IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	8985	9015	9015						
xcb_p	9532	9553	9553						
xcb_p	9533	9553/P	9553/P	9555					
xcb_p	9665	9694	9694						
xcb_p	9665	9695	9695						
xcb_p	9678	9684/P	9693	9694/P	9695/P	9702	9703/M	9712/P	9713
		9714	9715/M						
xcb_p	13375	13386	13386						
xcb_p	13376	13385	13386/P	13387	13392/S	13393/P	13395	13395	13396/S
		13403/P							
xcb_p	13428	13464	13464						
xcb_p	13455	13461/M	13462/P	13464/P	13464/P	13465/P	13466/P	13467/P	
xijl_ordinal	7271	7279/M	7282/M	7286					
xijl_ordinal	9665	9686/M	9686/M	9686					
xmcount	9945	10071/M							
xp	5306	8895	9009	9010	9018	9553/P	9555	10900	10905
		10905	10906	11066	11067	11327	11328/P	11336/M	11336
		11350	11354	11356	11358	11999/M	13387	13464/P	13465/P
		13466/P	13467/P						
xp	5492	8772	8826	8936	8962	11510			
xpfti	7687	7702/M							
xpfti	7955	7974/M							
xpfti	9775	9851/M	9852/S	9853/S	9854/S	9870/M	9871/S	9872/S	9873/S
xpfti	9942	9972/M	9988/M						
xpfti	9942	9976/M	10063/M						
xpfti	11313	11911/M	11912/S	11913/S	11914/P				
xpfti	12026	12317/M							
xrcount	9946	10070/M							
xsfid	7272	7274/M	7275/M	7278/M	7281/M	7285			
xsfid	9665	9686/M	9686/M	9686/M	9686/M	9686			
xsva	8875	8974/M							
xsva	10777	10810							
xsva	11312	11899/M	11901/M	11905	11905	11906/M	11906	11907	
xsva	12058	12107/M	12164	12181/M	12181	12216/M	12218	12237/P	12243/M
		12243	12265/P	12270	12302/M	12302			

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


```
          2 MODULE mmm$pfti_manager;
          3
          4 (
          5 (  PURPOSE: Memory_Manager
          6 (     This module contains the monitor routines that are used to
          7 (     locate all page frames bedlonging to a specified range of an SVA.
          8 (
          9
o       210 ( This decks defines compile-time constants to control conditional compilation
o       211 ( of debug code in memory manager modules in monitor mode. All constants should
o       212 ( be set to FALSE for the transmitted version of this deck.
o       213
o       214   CONST
o       215     mmc$debug = TRUE;
o       216
o       217
o       218   ?VAR
o       219      mmc$debug_check_queues: boolean := TRUE, {Check PQL linkage}
o       220      mmc$debug_relink_swapping_job: boolean := TRUE , {Stop if relink page of swapping job}
o       221      mmc$debug_pt: boolean := TRUE, {check PT - AST linkage}
o       222      mmc$debug_free_asid: boolean := TRUE, {Verify no attempt is made to free an AST
o       223                                             { entry with pages not in AVAIL queue.
o       224      mmc$debug_aste_p_from_pfti: boolean := TRUE, {Verify aste_p in pfti is correct}
o       225      mmc$debug_rma_list: boolean := TRUE, {Verify RMA list on lock/unlock rma list}
o       226      mmc$debug_esc_alloc: boolean := TRUE, {check for escaped allocation}
o       227      mmc$debug_ast_pft: boolean := TRUE?; {check pft.aste_p on reference to AST}
o       228


o       230 {External procedures used by this module.
o       231 { Try to keep them in alphabetical order.
o       232   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
o       233     VAR ijle_p: ^jmt$initiated_job_list_entry);
o       234
         1173
         1174 {  PURPOSE:     procedure mtp$error_stop
         1175 {    Prefixes 'ERR=VEOS1000-' to the string and calls mtp$step_unstep_system to write string and step system}
         1176
         1177   PROCEDURE [XREF] mtp$error_stop (text: string(*<=63) );
         1178
```

Global Variable Declarations - XREF and XDCL

```
1180 {Define pointer to Initiated Job List (IJL).
1181
1182   VAR
1183     jmv$ijl_p: [XREF]  jmt$ijl_p;
1211 {Pointer to the 'PAGE FRAME TABLE' (PFT)
1212
1213   VAR
1214     mmv$pft_p: [XREF] ^mmt$page_frame_table;
1215


1219 {Define page table length in words.
1220
1221   VAR
1222     mmv$pt_length: [XREF] integer;
1223


1225 {Pointer to the system PAGE TABLE (PT).
1226
1227   VAR
1228     mmv$pt_p: [XREF] ^ost$page_table;
1229
1232 {System page size.}
1233
1234   VAR
1235     osv$page_size: [XREF] ost$page_size;
1236
1250
1251   VAR
1252     mmv$pfti_array_p: [XDCL, #GATE] ^mmt$pfti_array;
1253
```


[XDCL] mmp$initialize_find_next_pfti

```
1255 {-------------------------------------------------------------------------------------------------------------
1256 {
1257 { The following procedures are used for locating pages belonging to a specified SVA to SVA+LENGTH range.
1258 { The procedures use an array (MMV$PFTI_ARRAY) allocated at deadstart time to contain the list of PFTIs.
1259 {       mmp$initialize_find_next_pfti    - fills the array with the list of PFTIs and returns the first one.
1260 {       mmp$find_next_pfti               - returns the next PFTI from the array. (0 = end of array)
1261 {       mmp$delete_last_pfti_from_array - deletes the most recently returned PFTI from the array. If the
1262 {                                         PFTI array is rescanned the entry will not be returned again;
1263 {       mmp$reset_find_next_pfti         - resets the array index so that subsequent calls to
1264 {                                         mmp$find_next_pfti will rescan the array and return all PFTIs.
1265 {
1266 { Depending on initialization options the PFTIs selected by theses routines are be restricted as follows:
1267 {       psc_nominal_queue    - only PFTIs of pages that are 'valid' will be returned (eg. working_set, wired,
1268 {                              fixed, shared, io_error)
1269 {       psc_all_except_avail- PFTI's of all pages in memory EXCEPT for pages in the AVAIL queue are returned.
1270 {                             PROCEDURES THAT CALL THESES ROUTINES MUST BE PREPARED TO HANDLE NOT FINDING
1271 {                             PAGES THAT ARE IN THE AVAIL QUEUE.
1272 {       psc_all              - the PFTI's of all pages within the range that are in memory are returned.  To
1273 {                              ensure that all pages of the segment are found, THE CALLER MUST SPECIFY THAT
1274 {                              THE ENTIRE SEGMENT BE SEARCHED.
1275 {-------------------------------------------------------------------------------------------------------------
1276
1277   PROCEDURE [XDCL] mmp$initialize_find_next_pfti
1278     (     xsva: ost$system_virtual_address;
1279           length: ost$segment_length;
1280           end_point_option: (include_partial_pages, exclude_partial_pages);
1281           page_selection_criteria: mmt$page_selection_criteria;
1282           aste_p: ^mmt$active_segment_table_entry;
1283      VAR xpfti: mmt$page_frame_index];
1284
1285     VAR
1286       found: boolean,
1287       hcount: 1 .. 32,
1288       high_offset: integer,
1289       ipti: integer,
1290       low_offset: integer,
1291       offset: integer,
1292       page_count: integer,
1293       pages_in_memory: integer,
1294       pfti: mmt$page_frame_index,
1295       pfti_index: integer,
1296       selection_page_queues: mmt$page_queue_set,
1297       sva: ost$system_virtual_address;
1298
1299     mmv$pfti_array_p^.pftis [0] := 0;
1300     pfti_index := 0;
1301     pages_in_memory := aste_p^.pages_in_memory;
1302
1303
1304 { Set up the parameters used to control the search. This includes the starting offset.
1305 { Determine the set of page queues a page can be linked to based on the page_selection_criteria.
1306
1307     IF (pages_in_memory > 0) AND (length > 0) THEN
1308       sva := xsva;
1309       offset := sva.offset;
1310       IF end_point_option = include_partial_pages THEN
```

[XDCL] mmp$initialize_find_next_pfti

```
3E  1311          page_count := (offset + length - 1) DIV osv$page_size - (offset DIV osv$page_size) + 1;
5A  1312        ELSE
5A  1313          page_count := (offset + length) DIV osv$page_size;
5A  1314          offset := ((offset + osv$page_size - 1) DIV osv$page_size) * osv$page_size;
5A  1315          page_count := page_count - offset DIV osv$page_size;
7C  1316        IFEND;
7C  1317        low_offset := offset - offset MOD osv$page_size;
7C  1318        high_offset := low_offset + page_count * osv$page_size - 1;
7C  1319
7C  1320        selection_page_queues := $mmt$page_queue_set [];
7C  1321        IF page_selection_criteria = psc_all THEN
A2  1322          selection_page_queues := -$mmt$page_queue_set [];
AC  1323        ELSEIF page_selection_criteria = psc_all_except_avail THEN
B2  1324          selection_page_queues := -$mmt$page_queue_set [mmc$pq_avail];
C0  1325        ELSE  {page_selection_criteria = psc_nominal_queue}
C0  1326          IF aste_p^.queue_id = mmc$pq_job_working_set THEN
CC  1327            selection_page_queues := $mmt$page_queue_set [mmc$pq_shared_io_error, mmc$pq_job_io_error,
D4  1328                  mmc$pq_job_working_set]; { Unwritable permanent file pages go to the shared_io_error q. }
D4  1329          ELSEIF ((aste_p^.queue_id >= mmc$pq_shared_first) AND (aste_p^.queue_id <= mmc$pq_shared_last)) THEN
E2  1330            selection_page_queues := -$mmt$page_queue_set [mmc$pq_free, mmc$pq_avail, mmc$pq_avail_modified,
F4  1331                  mmc$pq_wired, mmc$pq_job_fixed, mmc$pq_job_working_set, mmc$pq_job_io_error];
F4  1332          ELSEIF aste_p^.queue_id = mmc$pq_wired THEN
FE  1333            selection_page_queues := $mmt$page_queue_set [mmc$pq_wired];
106 1334          ELSEIF aste_p^.queue_id = mmc$pq_job_fixed THEN
10E 1335            selection_page_queues := $mmt$page_queue_set [mmc$pq_job_fixed];
114 1336          ELSE
114 1337            mtp$error_stop ('FIND NEXT PFTI -- BAD QUEUE ID');
134 1338          IFEND;
134 1339        IFEND;
134 1340
134 1341 { Search for pages.
134 1342
134 1343        IF (low_offset = 0) AND (length > 7ffffe00(16)) THEN
144 1344          pfti := aste_p^.pft_link.fwd;
144 1345          IF (page_selection_criteria = psc_all) THEN
150 1346
150 1347 { The request is for all the pages of a segment that are in memory, so use the search algorithm that uses
150 1348 { the segment/page frame link to find all pages of the segment.
150 1349
150 1350            WHILE pfti <> 0 DO
154 1351              mmv$pfti_array_p^.pftis [pfti_index] := pfti;
154 1352              pfti_index := pfti_index + 1;
154 1353              pfti := mmv$pft_p^ [pfti].segment_link.fwd;
154 1354            WHILEND;
18A 1355          ELSE
18A 1356
18A 1357 { The request length is for the entire segment, but not all queues should be searched.  Use the
18A 1358 { the segment/page frame link to find all pages of the segment and check if each page fits the
18A 1359 { page_selection_criteria.
18A 1360
18A 1361            WHILE (pfti <> 0) DO
192 1362              IF mmv$pft_p^ [pfti].queue_id IN selection_page_queues THEN
1BA 1363                mmv$pfti_array_p^.pftis [pfti_index] := pfti;
1BA 1364                pfti_index := pfti_index + 1;
1C8 1365              IFEND;
1C8 1366              pfti := mmv$pft_p^ [pfti].segment_link.fwd;
```


[XDCL] mmp$initialize_find_next_pfti

```
1C8 1367            WHILEND;
1E8 1368          IFEND;
1EC 1369
1EC 1370        ELSEIF (page_count < pages_in_memory DIV 4) THEN
1F6 1371
1F6 1372 { If the request is for a small percentage of the pages in memory, use the search algorith that uses
1F6 1373 { the #HASH instruction to locate pages. This algorithm is NOT efficient for searches that look for
1F6 1374 { a large number of pages, because it does a HASH on each page in the range.  If the page is found,
1F6 1375 { it is put in the pfti array only if the page satisfies the page selection criteria.
1F6 1376
1F6 1377          WHILE page_count > 0 DO
1FA 1378            sva.offset := offset;
1FA 1379            #HASH_SVA (sva, ipti, hcount, found);
200 1380            page_count := page_count - 1;
200 1381            offset := offset + osv$page_size; {Must be integer to prevent end case at end of segment (2**31)}
200 1382            IF found THEN
21C 1383              pfti := (mmv$pt_p^ [ipti].rma * 512) DIV osv$page_size;
21C 1384              IF mmv$pft_p^ [pfti].queue_id IN selection_page_queues THEN
262 1385                mmv$pfti_array_p^.pftis [pfti_index] := pfti;
262 1386                pfti_index := pfti_index + 1;
270 1387              IFEND;
270 1388              pages_in_memory := pages_in_memory - 1;
270 1389              IF pages_in_memory = 0 THEN
276 1390                page_count := 0;
27A 1391              IFEND;
27A 1392            IFEND;
27A 1393          WHILEND;
282 1394
282 1395        ELSE
282 1396
282 1397 { The request is for a majority of the pages in memory.  Again, use the search algorithm that uses the
282 1398 { segment/page frame link to find all pages of the segment.  We need to check if the page offset is within
282 1399 { the range requested and decide if the page satisfies the selection criteria.
282 1400
282 1401          pfti := aste_p^.pft_link.fwd;
282 1402          WHILE (pfti <> 0) AND (page_count > 0) DO
292 1403            IF (mmv$pft_p^ [pfti].sva.offset >= low_offset) AND
2BC 1404                  (mmv$pft_p^ [pfti].sva.offset <= high_offset) THEN
2BC 1405              page_count := page_count - 1;
2BC 1406              IF mmv$pft_p^ [pfti].queue_id IN selection_page_queues THEN
2CE 1407                mmv$pfti_array_p^.pftis [pfti_index] := pfti;
2CE 1408                pfti_index := pfti_index + 1;
2DC 1409              IFEND;
2DC 1410            IFEND;
2DC 1411            pfti := mmv$pft_p^ [pfti].segment_link.fwd;
2DC 1412          WHILEND;
300 1413
300 1414        IFEND;
300 1415      IFEND;  { pages_in_memory > 0 }
300 1416
300 1417      mmv$pfti_array_p^.pfti_first := 0;
300 1418      mmv$pfti_array_p^.pfti_index := 0;
300 1419      mmv$pfti_array_p^.last_pfti_index := pfti_index;
300 1420      mmv$pfti_array_p^.pftis [pfti_index] := 0;
300 1421      xpfti := mmv$pfti_array_p^.pftis [0];
300 1422
```

[XDCL] mmp$initialize_find_next_pfti

```
300  1423   PROCEND mmp$initialize_find_next_pfti;
  o  1424
  o  1425
  o  1426
```


INLINE Procedures used in other modules

```
          1429
          1430   PROCEDURE [INLINE] mmp$delete_last_pfti_from_array;


      o   1437
      o   1438   PROCEDURE [INLINE] mmp$fetch_pfti_array_size
      o   1439     (VAR pfti_size: integer);
      o   1440


          1447
          1448   PROCEDURE [INLINE] mmp$find_next_pfti
          1449     (VAR xpfti: mmt$page_frame_index);
          1450


      o   1469
      o   1470   PROCEDURE [INLINE] mmp$reset_find_next_pfti
      o   1471     (VAR xpfti: mmt$page_frame_index);
      o   1472


          1496
          1497   PROCEDURE [INLINE] mmp$reset_store_pfti;
          1498


      o   1506
      o   1507   PROCEDURE [INLINE] mmp$store_pfti (pfti: mmt$page_frame_index);
      o   1508


          1516
          1517   PROCEDURE [INLINE] mmp$reset_store_pfti_reverse;
          1518


      o   1526
      o   1527   PROCEDURE [INLINE] mmp$store_pfti_reverse (pfti: mmt$page_frame_index);
      o   1528
      o   1537 MODEND mmm$pfti_manager
```

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

**** NO DIAGNOSTICS

IDENTIFIER	DEFINED ON LINE	REFERENCES							
aste_p	1282	1301	1326	1329	1329	1332	1334	1344	1401
block_index	180	236/S							
block_number	179	236/S							
block_p	1189	236							
dfc$command_record_bytes	356	364							
dfc$division_overwrite_words	343	371							
dfc$esm_command_record_size	364	372							
dfc$esm_header_record_size	365	372							
dfc$esm_maintenance_buf_size	344	375							
dfc$esm_memory_base_shift	350	372	373	373					
dfc$header_record_bytes	355	365							
dfc$max_esm_memory_size	345	374							
dfc$max_number_of_mainframes	352	337							
dfc$min_data_record_bytes	360	371							
dfc$min_esm_division_size	370	374							
dft$mainframe_set	337	287	288	446	447				
dmt$system_file_id	386	319							
end_point_option	1280	1310							
found	1286	1379	1382						
fwd	196	1344	1353	1366	1401	1411			
gft$file_descriptor_index	162	152							
gft$system_file_identifier	151	141	386						
gft$table_residence	165	153							
hcount	1287	1379							
high_offset	1288	1318/M	1404						
ijl_ordinal	232	236/S	236/S						
ijle_p	233	236/M							
include_partial_pages	1280	1310							
index_p	1202	236							
iot$io_error	1131	320	1085						
ipti	1289	1379	1383/S						
jmc$highest_service_accumulator	834	835							
jmc$ies_job_swapped	614	623							
jmc$ies_swapin_in_progress	613	622							
jmc$iss_idle_tasks_initiated	629	656							
jmc$iss_swapin_io_complete	654	657							
jmc$iss_swapin_requested	650	657							
jmc$iss_swapout_complete	649	656							
jmc$iss_swapped_io_cannot_init	640	667							
jmc$iss_swapped_no_io	631	666							
jmc$kjl_maximum_entries	407	400	401	786					
jmc$kol_maximum_entries	417	402							
jmc$max_ajl_ord	399	392							
jmc$max_dispatching_control	573	577							
jmc$max_dispatching_priority	495	455	458	459					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$max_ijl_index_count	188	1200							
jmc$maximum_job_classes	764	767							
jmc$maximum_job_count	414	407							
jmc$maximum_output_count	424	417							
jmc$maximum_service_classes	867	870							
jmc$min_dispatching_control	572	576							
jmc$null_service_class	860	861							
jmc$priority_p1	509	456							
jmc$priority_p10	518	457							
jmc$priority_p14	522	457							
jmc$priority_p8	516	456							
jmc$reserved_ajls	403	398							
jmc$service_accumulator_maximum	826	823							
jmc$system_default_offset	850	851							
jmc$system_supplied_name_size	995	992							
jmc$unlimited_offset	847	836							
jmp$get_ijle_p	232	238							
jmt$ajl_ordinal	392	256							
jmt$delayed_swapin_work	439	286	443						
jmt$dispatching_control_index	576	533	543						
jmt$dispatching_controls	546	544							
jmt$dispatching_priority	455	268	534	535	536	548			
jmt$ijl_block_index	184	180	1202						
jmt$ijl_block_number	183	179	1190	1191					
jmt$ijl_dispatching_control	532	269							
jmt$ijl_entry_status	609	255							
jmt$ijl_ordinal	178	134	232	275	303	1010	1011	1077	
jmt$ijl_p	1188	1183							
jmt$ijl_page_fault_count	683	678	679	680					
jmt$ijl_page_stats	677	673							
jmt$ijl_service_class_stats	671	290							
jmt$ijl_statistics	716	289							
jmt$ijl_swap_count	692	688	689						
jmt$ijl_swap_counts	687	309	674						
jmt$ijl_swap_status	627	258	259	260					
jmt$initiated_job_list_block	1199	1205							
jmt$initiated_job_list_entry	252	233	1035	1202					
jmt$initiated_job_list_p	1205	1189							
jmt$input_file_location	806	801							
jmt$job_abort_disposition	815	799							
jmt$job_class	767	314							
jmt$job_mode	770	271							
jmt$job_priority	775	311	312						
jmt$job_recovery_disposition	818	800							
jmt$kjl_index	786	257							
jmt$queue_file_ijl_information	798	296							
jmt$scheduling_data	302	280							
jmt$service_accumulator	823	304	305	306					
jmt$service_class_index	870	315							
jmt$swap_data	318	282							
jmt$swapout_reasons	873	310							
jmt$swapped_job_entry	888	327	1036						
jmt$system_supplied_name	992	253							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$task_time_slice	586	566	567						
jmt$time_slice_values	585	550							
jmv$ijl_p	1183	236							
jsc$isqi_swapped_io_completed	1015	1017							
jsc$isqi_swapped_io_not_init	1014	1017							
jst$changed_asid_entry	1058	1049							
jst$ijl_swap_queue_id	1014	1009							
jst$ijl_swap_queue_link	1008	264							
jst$io_control_information	1022	283							
jst$swap_file_descriptor	1034	284							
jst$swapped_page_descriptor	1043	1041							
jst$swapped_page_descriptors	1040	1037							
last_pfti_index	17	1419/M	1442	1456	1457	1461	1481	1482	1487
		1500/M	1510/S	1511/M	1511	1520/M			
length	1279	1307	1311	1313	1343				
low_offset	1290	1317/M	1318	1343	1403				
mmc$pq_avail	60	106	1324	1330					
mmc$pq_avail_modified	61	1330							
mmc$pq_free	59	118	1330						
mmc$pq_job_fixed	100	107	119	1331	1334	1335			
mmc$pq_job_io_error	101	1327	1331						
mmc$pq_job_working_set	102	119	120	1326	1328	1331			
mmc$pq_shared_first	108	1329							
mmc$pq_shared_first_site	110	114							
mmc$pq_shared_io_error	97	1327							
mmc$pq_shared_last	115	1329							
mmc$pq_shared_num_sites	111	114							
mmc$pq_shared_other	69	109							
mmc$pq_shared_site_01	71	110							
mmc$pq_shared_site_25	95	115							
mmc$pq_shared_task_service	64	108							
mmc$pq_swapped_io_error	98	118							
mmc$pq_wired	62	105	1331	1332	1333				
mmp$delete_last_pfti_from_array	1430	1434							
mmp$fetch_pfti_array_size	1438	1444							
mmp$find_next_pfti	1448	1466							
mmp$initialize_find_next_pfti	1277	1423							
mmp$reset_find_next_pfti	1470	1493							
mmt$active_segment_table_entry	131	147	1046	1084	1282				
mmt$ast_index	1067	326	1061						
mmt$global_page_queue_index	118	1169							
mmt$global_page_queue_list_ent	1159	1169							
mmt$job_page_queue_index	119	890	1170						
mmt$job_page_queue_list	1170	281							
mmt$link	194	132	1074	1075	1156				
mmt$locked_page	1096	1080							
mmt$memory_reserve_request	1137	274							
mmt$page_age	1103	1083	1107	1107					
mmt$page_frame_index	21	18	196	196	1023	1025	1026	1027	1139
		1140	1283	1294	1449	1453	1471	1475	1507
		1527							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$page_frame_queue_id	120	48	140	1024	1078				
mmt$page_frame_table	1089	1214							
mmt$page_frame_table_entry	1073	1044	1089						
mmt$page_queue_list_entry	1155	1160	1170						
mmt$page_queue_set	48	1296	1320	1322	1324	1327	1330	1333	1335
mmt$page_selection_criteria	125	1281							
mmt$pfti_array	14	1252							
mmv$pft_p	1214	1353	1362	1366	1384	1403	1404	1406	1411
mmv$pfti_array_p	1252	1299/M	1351/M	1363/M	1385/M	1407/M	1417/M	1418/M	1419/M
		1420/M	1421	1432/S	1432/M	1442	1442	1456	1456
		1457	1457	1458/M	1458	1459	1459/S	1461	1461
		1478/M	1478	1479	1479/S	1481	1481	1482	1482
		1483/M	1483	1484	1484/S	1487	1487	1500/M	1501/M
		1502/M	1510/S	1510/M	1511/M	1511	1520/M	1520	1521/M
		1521	1522/M	1522	1530/M	1530	1531/M	1531	1532/S
		1532/M							
mmv$pt_p	1228	1383							
mtp$error_stop	1177	1337							
offset	907	1309	1378/M	1403	1404				
offset	1291	1309/M	1311	1311	1313	1314/M	1314	1315	1317
		1317	1378	1381/M	1381				
osc$free_running_clock_maximum	206	203							
osc$invalid_ring	924	964							
osc$max_page_frames	25	15	16	17	21	133	321	322	889
		891	1157	1163					
osc$max_page_size	1247	1243							
osc$max_page_table_entries	26	29							
osc$max_ring	923	964	965						
osc$max_segment_length	947	970							
osc$max_tasks	1129	1126							
osc$maximum_offset	946	947	967	967	968				
osc$maximum_segment	945	966							
osc$min_page_size	1246	1243							
osc$min_ring	922	965							
osc$task_time_slice_maximum	597	600							
ost$asid	910	138	265	906	1048	1059	1060		
ost$cp_time	704	672	717						
ost$cp_time_value	702	307	705	706					
ost$free_running_clock	203	137	276	277	278	279	313	323	324
		325	537	549					
ost$global_task_id	1120	270	299						
ost$key_lock_value	959	956							
ost$page_id	31	41							
ost$page_size	1243	1235							
ost$page_table	45	1228							
ost$page_table_entry	36	45	1045						
ost$page_table_index	29	45	1081						
ost$paging_statistics	740	718							
ost$ring	964	976							
ost$segment	966	977							
ost$segment_length	970	1279							
ost$segment_offset	967	907	978						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$system_virtual_address	905	1086	1278	1297					
ost$task_index	1126	1114	1115	1121					
ost$task_time_slice	600	586							
osv$page_size	1235	1311	1311	1313	1314	1314	1314	1315	1317
		1318	1381	1383					
page_count	1292	1311/M	1313/M	1315/M	1315	1318	1370	1377	1377
		1380/M	1380	1390/M	1402	1402	1405/M	1405	
page_selection_criteria	1281	1321	1323	1345					
pages_in_memory	133	1301							
pages_in_memory	1293	1301/M	1307	1370	1388/M	1388	1389		
pft_link	132	1344	1401						
pfti	1294	1344/M	1350	1350	1351	1353/M	1353/S	1361	1361
		1362/S	1363	1366/M	1366/S	1383/M	1384/S	1385	1401/M
		1402	1402	1403/S	1404/S	1406/S	1407	1411/M	1411/S
pfti	1453	1455/M	1456	1456	1459/M	1462/M	1464		
pfti	1475	1479/M	1481	1481	1484/M	1488/M	1490		
pfti	1507	1510							
pfti	1527	1532							
pfti_first	15	1417/M	1442	1478	1479/S	1502/M	1522/M	1530	1531/M
		1531	1532/S						
pfti_index	16	1418/M	1432/S	1456	1457	1458/M	1458	1459/S	1461
		1478/M	1481	1482	1483/M	1483	1484/S	1487	1501/M
		1521/M	1530/M						
pfti_index	1295	1300/M	1351/S	1352/M	1352	1363/S	1364/M	1364	1385/S
		1386/M	1386	1407/S	1408/M	1408	1419	1420/S	
pfti_size	1439	1442/M							
pftis	18	1299/M	1351/M	1363/M	1385/M	1407/M	1420/M	1421	1432/M
		1459	1479	1484	1510/M	1520	1521	1522	1532/M
psc_all	125	1321	1345						
psc_all_except_avail	125	1323							
queue_id	140	1326	1329	1329	1332	1334			
queue_id	1078	1362	1384	1406					
rma	42	1383							
segment_link	1075	1353	1366	1411					
selection_page_queues	1296	1320/M	1322/M	1324/M	1327/M	1330/M	1333/M	1335/M	1362
		1384	1406						
sft$counter	750	719	720						
sva	1086	1403	1404						
sva	1297	1308/M	1309	1378/M	1379				
tmt$task_queue_link	1113	1082							
xpfti	1283	1421/M							
xpfti	1449	1464/M							
xpfti	1471	1490/M							
xsva	1278	1308							

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

MMM$SEGMENT_MANAGER_JOB_TEMP

```
          3 MODULE mmm$segment_manager_job_temp;
          4
    o  4524
    o  4525   VAR
    o  4526     gfv$null_sfid: [XREF, READ, OSS$MAINFRAME_WIRED_LITERAL] gft$system_file_identifier;
    o  4527
    o  4528
    o  4529   SECTION
    o  4530     oss$mainframe_wired_literal: READ;
    o  4531
    o  4532  VAR
    o  4533    mmv$file_allocation_interval: [XREF] integer;
    o  4534
    o  4535   VAR
    o  4536     mmv$shadow_by_segnum: [XREF] boolean;
    o  4537 {System page size.}
    o  4538
    o  4539   VAR
    o  4540     osv$page_size: [XREF] ost$page_size;
    o  4541
       4544
       4545   VAR
       4546     osv$task_private_heap: [XREF, READ, oss$job_pageable] ^ost$heap;
       4547
    o  4553
    o  4554 {  Define variables that are global to this module.
    o  4555
    o  4556   VAR
    o  4557     osv$system_privilege_map: [XDCL, #GATE, oss$task_private] ost$system_privilege_map :=
    o  4558           [REP (mmc$default_sdt_length + 1) of TRUE];
    o  4559
```

DETERMINE_VALIDATING_RING_NUM

```
o  4562
o  4563   PROCEDURE [INLINE] determine_validating_ring_num
o  4564     (    caller_ring: ost$ring;
o  4565          validation_ring_number: ost$valid_ring;
o  4566      VAR validating_ring_num: ost$valid_ring);
o  4567
o  4568     IF validation_ring_number < caller_ring THEN
o  4569       validating_ring_num := caller_ring;
o  4570     ELSE
o  4571       validating_ring_num := validation_ring_number;
o  4572     IFEND;
o  4573
o  4574   PROCEND determine_validating_ring_num;
o  4575
```


UPDATE_PASSIVE_WITH_ACTIVE

```
o  4577
o  4578   PROCEDURE update_passive_with_active
4  4579     (    segment_p: ^cell;
4  4580      VAR status: ost$status);
4  4581
4  4582     TYPE
4  4583       offset_list = array [1 .. * ] of ost$segment_offset;
4  4584
4  4585     VAR
4  4586       access_selections: mmt$access_selections,
4  4587       address_list: array [1 .. 100] of dmt$addr_length_pair,
4  4588       addr_returned: integer,
4  4589       dest_p: mmt$segment_pointer,
4  4590       dest: ^cell,
4  4591       dm_element_length: ost$segment_length,
4  4592       dm_element_offset: ost$segment_offset,
4  4593       fde_p: gft$file_desc_entry_p,
4  4594       file_limits_to_enforce: sft$file_space_limit_kind,
4  4595       i: integer,
4  4596       in_memory: boolean,
4  4597       list_overflow: boolean,
4  4598       list_p: ^offset_list,
4  4599       local_status: ost$status,
4  4600       memory_list_index: integer,
4  4601       offsets_returned: integer,
4  4602       sdt_p: ^mmt$segment_descriptor,
4  4603       sdtx_p: ^mmt$segment_descriptor_extended,
4  4604       segnum: ost$segment,
4  4605       source: ^cell,
4  4606       starting_addr: ost$segment_offset;
4  4607
4  4608 {  Get access selection before change to restore at end of proc.
4  4609
4  4610     mmp$validate_segment_number (#SEGMENT (segment_p), sdt_p, sdtx_p, status);
2E 4611     IF NOT status.normal THEN
36 4612       RETURN;
38 4613     IFEND;
38 4614
38 4615     IF sdtx_p^.software_attribute_set = $mmt$software_attribute_set
46 4616          [mmc$sa_read_transfer_unit, mmc$sa_free_behind] THEN
46 4617       access_selections := mmc$as_sequential;
4C 4618     ELSEIF sdtx_p^.software_attribute_set = $mmt$software_attribute_set [mmc$sa_read_transfer_unit] THEN
52 4619       access_selections := mmc$as_read_tu;
58 4620     ELSE
58 4621       access_selections := mmc$as_random;
5A 4622     IFEND;
5A 4623
5A 4624 {  Set read_transfer_unit attribute in ACTIVE segment.
5A 4625
5A 4626     mmp$set_access_selections (segment_p, mmc$as_sequential, status);
74 4627
74 4628 {  Obtain access to PASSIVE segment.
74 4629
74 4630     mmp$open_file_segment (sdtx_p^.shadow_info.shadow_sfid, NIL, mmc$cell_pointer, 1, sfc$no_limit, dest_p,
B6 4631          status);
B6 4632     IF NOT status.normal THEN
```

UPDATE_PASSIVE_WITH_ACTIVE

```
BE  4633        RETURN;
CO  4634      IFEND;
CO  4635
CO  4636      offsets_returned := 100;
CO  4637      REPEAT
C8  4638        ALLOCATE list_p: [1 .. offsets_returned] IN osv$task_private_heap^;
10A 4639        mmp$fetch_offset_modified_pages (segment_p, FALSE (return_unallocated_offsets) , list_p^,
13C 4640              offsets_returned, status);
13C 4641        IF NOT status.normal THEN
144 4642          FREE list_p IN osv$task_private_heap^;
16A 4643          RETURN;
172 4644        IFEND;
172 4645        IF offsets_returned > UPPERBOUND (list_p^) THEN
186 4646          FREE list_p IN osv$task_private_heap^;
1AE 4647        IFEND;
1AE 4648      UNTIL offsets_returned <= UPPERBOUND (list_p^);
1BE 4649
1BE 4650      IF offsets_returned <> 0 THEN
1C6 4651 {  Convert offsets to PVAs and move all modified pages from the ACTIVE segment
1C6 4652        FOR i := 1 TO offsets_returned DO
1CE 4653          source := #ADDRESS (1, #SEGMENT (segment_p), list_p^ [i]);
1CE 4654          dest := #ADDRESS (1, #SEGMENT (dest_p.cell_pointer), list_p^ [i]);
1CE 4655          i#move (source, dest, osv$page_size);
248 4656          mmp$advise_out (dest, osv$page_size, status);
266 4657        FOREND;
270 4658      IFEND;
270 4659
270 4660
270 4661 {  Determine whether the segment is assigned to a device.
270 4662 {  Get and move all initialized addresses from ACTIVE file.
270 4663 {  If the segment is not assigned, no further action is then required.
270 4664
270 4665
270 4666      starting_addr := 0;
270 4667      gfp$get_fde_p (sdtx_p^.sfid, fde_p);
2CA 4668      list_overflow := (fde_p^.media = gfc$fm_mass_storage_file);
2CA 4669
2CA 4670    /get_list_of_addresses/
2CA 4671      WHILE list_overflow DO
2EA 4672        dmp$get_initialized_addresses (sdtx_p^.sfid, starting_addr, address_list, addr_returned, list_overflow,
32C 4673              status);
32C 4674        IF NOT status.normal THEN
334 4675          RETURN;
336 4676        IFEND;
336 4677
336 4678        IF addr_returned <> 0 THEN
33E 4679          FOR i := 1 TO addr_returned DO
344 4680
344 4681 { Look for offsets which may have already been written. Specifically, we are looking for pages which are on
344 4682 { disk and are not modified in real memory at the same time.
344 4683
344 4684            dm_element_length := address_list [i].length;
344 4685            dm_element_offset := address_list [i].addr;
344 4686            memory_list_index := 1;
344 4687            WHILE (dm_element_length > 0) AND (memory_list_index > 0) DO
364 4688              memory_list_index := offsets_returned;
```


UPDATE_PASSIVE_WITH_ACTIVE

```
364 4689
364 4690 { Search the list (of offsets of all modified pages for the ACTIVE segment) for the equivalent offset returned
364 4691 { from Device Management.  If ANY of the pages in <addr + length> are not found, move the disk pages, starting
364 4692 { at <address_list [i].addr> and continuing through <address_list [i].addr + address_list [i].length.>.  In
364 4693 { this case we MAY overwrite some addresses which were processed in the previous update above, but this is OK.
364 4694
364 4695              WHILE (memory_list_index > 0) AND (list_p^ [memory_list_index] <> dm_element_offset) DO
37E 4696                memory_list_index := memory_list_index - 1;
37E 4697              WHILEND;
396 4698              dm_element_offset := dm_element_offset + osv$page_size;
396 4699              dm_element_length := dm_element_length - osv$page_size;
396 4700            WHILEND;
3A6 4701
3A6 4702            IF memory_list_index = 0 THEN
3AA 4703
3AA 4704 { The address offset returned from DM was not written (either partially or completely) in the previous
3AA 4705 { process; write it now.
3AA 4706
3AA 4707              source := #ADDRESS (1, #SEGMENT (segment_p), address_list [i].addr);
3AA 4708              dest := #ADDRESS (1, #SEGMENT (dest_p.cell_pointer), address_list [i].addr);
3AA 4709              i#move (source, dest, address_list [i].length);
41E 4710              mmp$advise_out (source, address_list [i].length, status);
446 4711              mmp$advise_out (dest, address_list [i].length, status);
464 4712            IFEND;
464 4713          FOREND;
468 4714          starting_addr := address_list [addr_returned].addr + address_list [addr_returned].length;
484 4715        IFEND;
484 4716      WHILEND /get_list_of_addresses/;
490 4717
490 4718      mmp$set_access_selections (segment_p, access_selections, status);
4A8 4719      mmp$close_segment (dest_p, 1, local_status);
4C8 4720      FREE list_p IN osv$task_private_heap^;
4F2 4721
4F2 4722    PROCEND update_passive_with_active;
```

MMP$CHANGE_SEGMENT_INHERITANCE

```
 0  4725 {
 0  4726 {   The purpose of this request is to change the segment inheritance of the
 0  4727 { specified segment. Only segments of inheritance mmc$si_none can be modified
 0  4728 { with this request.
 0  4729 {
 0  4730 {    MMP$CHANGE_SEGMENT_INHERITANCE (segment_pointer, segment_inheritance, status)
 0  4731 {
 0  4732 { SEGMENT_POINTER: (input) This parameter specifies the process_virtual_address of
 0  4733 {          the segment which is to be changed.
 0  4734 {
 0  4735 { SEGMENT_INHERITANCE: (input) This parameter specifies the new segment inheritance.
 0  4736 {          The only valid inheritances for this request are mmc$si_share_segment and
 0  4737 {          mmc$si_transfer_segment.
 0  4738 {
 0  4739 { STATUS: (output) This parameter specifies the request status.
 0  4740 {
 0  4741
 0  4742   PROCEDURE [XDCL] mmp$change_segment_inheritance
 0  4743     (     pva: ^cell;
 0  4744           segment_inheritance: mmt$segment_inheritance;
 0  4745      VAR status: ost$status);
 0  4746
 0  4747     VAR
 0  4748       caller_id: ost$caller_identifier,
 0  4749       r1_status: ost$status,
 0  4750       validating_ring_number: ost$valid_ring;
 0  4751
 0  4752     status.normal := TRUE;
 C  4753     #CALLER_ID (caller_id);
 C  4754     determine_validating_ring_num (caller_id.ring, #RING (pva), validating_ring_number);
3C  4755     mmp$change_seg_inheritance_r1 (#SEGMENT (pva), validating_ring_number, segment_inheritance, r1_status);
6E  4756     IF NOT r1_status.normal THEN
76  4757       osp$set_status_condition (r1_status.condition, status);
88  4758     IFEND;
88  4759
88  4760   PROCEND mmp$change_segment_inheritance;
 0  4761
```


MMP$CHANGE_SEGMENT_NUMBER

```
 0  4764 {
 0  4765 {  The purpose of this request is to associate a previously opened
 0  4766 { file with a different segment number. The old segment is deleted
 0  4767 { when the new segment number has been assigned.
 0  4768 {
 0  4769 {     MMP$CHANGE_SEGMENT_NUMBER (segment_pointer, segment_number,
 0  4770 {       validation_ring_number, new_segment_pointer, status);
 0  4771 {
 0  4772 { Segment_pointer: (INPUT) This parameter specifies the process virtual
 0  4773 {    address of the segment.
 0  4774 {
 0  4775 { Segment_number: (INPUT) This parameter is the new segment number
 0  4776 {    to be associated with the file.
 0  4777 {
 0  4778 { Validation_ring_number: (INPUT) This parameter specifies the ring
 0  4779 {    of execution for whom the segment is being modified. The ring
 0  4780 {    number used for validation is the maximum of the caller ring
 0  4781 {    number and the 'validation_ring_number'.
 0  4782 {
 0  4783 { New_segment_pointer: (OUTPUT) This parameter specifies the process
 0  4784 {    virtual address of the file with the new segment number.
 0  4785 {
 0  4786 { Status: (OUTPUT) This parameter specifies the request status.
 0  4787 {    CONDITIONS:
 0  4788 {         mme$ring_violation
 0  4789 {         mme$unsupported_segment_kind
 0  4790 {    IDENTIFIER: 'MM'
 0  4791 {
 0  4792
 0  4793
 0  4794   PROCEDURE [XDCL] mmp$change_segment_number
 0  4795     (     segment_pointer: amt$segment_pointer;
 0  4796           segment_number: ost$segment;
 0  4797           validation_ring_number: ost$valid_ring;
 0  4798      VAR new_segment_pointer: amt$segment_pointer;
 0  4799      VAR status: ost$status);
 0  4800
 0  4801     VAR
 0  4802       caller_id: ost$caller_identifier,
 0  4803       r1_segment_pointer: amt$segment_pointer,
 0  4804       r1_status: ost$status,
 0  4805       validating_ring_number: ost$valid_ring;
 0  4806
 0  4807     status.normal := TRUE;
 C  4808     #CALLER_ID (caller_id);
 C  4809     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
36  4810     mmp$change_segment_number_r1 (#SEGMENT (segment_pointer.cell_pointer), segment_number,
6C  4811           validating_ring_number, r1_status);
6C  4812     IF r1_status.normal THEN
74  4813       new_segment_pointer := segment_pointer;
82  4814       new_segment_pointer.cell_pointer := #ADDRESS (#RING (segment_pointer.cell_pointer),
9E  4815             segment_number, #OFFSET (segment_pointer.cell_pointer));
9E  4816     ELSE
9E  4817       osp$set_status_condition (r1_status.condition, status);
B0  4818     IFEND;
B0  4819
```

MMP$CHANGE_SEGMENT_NUMBER

```
BO  4820   PROCEND mmp$change_segment_number;
 O  4821
```


MMP$CHANGE_STACK_ATTRIBUTE

```
 O  4823
 O  4824 {  The purpose of this request is to modify the stack attribute
 O  4825 {  which determines whether or not the pages beyond the current
 O  4826 {  top-of-stack pointer are freed. Only the stack of the caller ring
 O  4827 {  can be modified by this request.
 O  4828 {
 O  4829 {  MMP$CHANGE_STACK_ATTRIBUTE (stack_pages_to_be_freed, status);
 O  4830 {
 O  4831 {    STACK_PAGES_TO_BE_FREED: (INPUT, BOOLEAN) This parameter specifies
 O  4832 {       whether or not the pages beyond the current top-of-stack are freed.
 O  4833 {
 O  4834 {    STATUS: (OUTPUT, OST$STATUS) The request status is returned in this
 O  4835 {    parameter.
 O  4836 {
 O  4837
 O  4838    PROCEDURE [XDCL, #GATE] mmp$change_stack_attribute
 O  4839      (     stack_pages_to_be_freed: boolean;
 O  4840       VAR status: ost$status);
 O  4841
 O  4842      VAR
 O  4843        caller_id: ost$caller_identifier,
 O  4844        r1_status: ost$status;
 O  4845
 O  4846      status.normal := TRUE;
 C  4847      #CALLER_ID (caller_id);
 C  4848
 C  4849      mmp$change_stack_attribute_r1 (stack_pages_to_be_freed, caller_id.ring, r1_status);
42  4850      IF NOT r1_status.normal THEN
4A  4851        osp$set_status_condition (r1_status.condition, status);
5C  4852      IFEND;
5C  4853
5C  4854    PROCEND mmp$change_stack_attribute;
 O  4855
```

MMP$CLOSE_SEGMENT

```
        o  4857 {
        o  4858 {   The purpose of this request is to close/delete a segment.  If the
        o  4859 { caller is not within the read bracket of the segment being closed the
        o  4860 { call is rejected.
        o  4861 {
        o  4862 {       MMP$CLOSE_SEGMENT (POINTER, VALIDATION_RING_NUMBER, STATUS)
        o  4863 {
        o  4864 { POINTER: (input_output) This parameter specifies the segment
        o  4865 {       to be deleted.  This pointer will be set to NIL when the
        o  4866 {       request completes.
        o  4867 {
        o  4868 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring of
        o  4869 {       execution for whom the segment is being closed.  The ring
        o  4870 {       number used for validation is the max of the caller ring number
        o  4871 {       and 'validation_ring_number'.
        o  4872 {
        o  4873 { STATUS: (output) This parameter specifies the request status.
        o  4874 {       The possible error codes are:
        o  4875 {            dme$file_descriptor_not_deleted
        o  4876 {            mme$invalid_close_segment_req
        o  4877 {            mme$invalid_shared_taskid
        o  4878 {            mme$segment_number_not_in_use
        o  4879 {            mme$segment_number_too_big
        o  4880 {
        o  4881
        o  4882   PROCEDURE [XDCL, #GATE] mmp$close_segment
        o  4883     (VAR pointer: mmt$segment_pointer;
        o  4884          validation_ring_number: ost$valid_ring;
        o  4885      VAR status: ost$status);
        o  4886
        o  4887     VAR
        o  4888       caller_id: ost$caller_identifier,
        o  4889       r1_status: ost$status,
        o  4890       validating_ring_number: ost$valid_ring;
        o  4891
        o  4892     #CALLER_ID (caller_id);
        C  4893     #KEYPOINT (osk$entry, osk$m * #SEGMENT (pointer.cell_pointer), mmk$close);
       2A  4894     status.normal := TRUE;
       2A  4895     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
       4C  4896     mmp$invalidate_segment (#SEGMENT (pointer.cell_pointer), validating_ring_number,
       86  4897          NIL (shared_taskid_array), r1_status);
       86  4898     IF NOT r1_status.normal THEN
       8E  4899       osp$set_status_condition (r1_status.condition, status);
       A4  4900     ELSE
       A4  4901       pointer.cell_pointer := NIL;
       A8  4902     IFEND;
       A8  4903     #KEYPOINT (osk$exit, 0, mmk$close);
       AC  4904
       AC  4905   PROCEND mmp$close_segment;
```


MMP$CLOSE_SHARED_STACK

```
        o  4907 {
        o  4908 {     The purpose of this request is to close/delete a stack segment
        o  4909 { being used by ADA tasks.
        o  4910 {
        o  4911 { MMP$CLOSE_SHARED_STACK (POINTER, SHARED_TASKID_ARRAY, STATUS)
        o  4912 {
        o  4913 { POINTER: (INPUT/OUTPUT) This parameter specifes the segment to
        o  4914 {                         be deleted. This pointer will be set to
        o  4915 {                         NIL when the request completes.
        o  4916 {
        o  4917 { SHARED_TASKID_ARRAY: (INPUT) This paramter specifies the array
        o  4918 {                             of taskids for the tasks in which
        o  4919 {                             this segment is being closed/deleted.
        o  4920 {
        o  4921 { STATUS: (OUTPUT) This parameter specifies the request status.
        o  4922 {
        o  4923
        o  4924   PROCEDURE [XDCL] mmp$close_shared_stack
        o  4925     (VAR pointer: mmt$segment_pointer;
        o  4926        , shared_taskid_array: ^array [1 .. * ] of pmt$task_id;
        o  4927      VAR status: ost$status);
        o  4928
        o  4929     VAR
        o  4930       caller_id: ost$caller_identifier,
        o  4931       r1_status: ost$status;
        o  4932
        o  4933     #CALLER_ID (caller_id);
        C  4934     status.normal := TRUE;
        C  4935     mmp$invalidate_segment (#SEGMENT (pointer.cell_pointer), caller_id.ring, shared_taskid_array, r1_status);
       54  4936     IF NOT r1_status.normal THEN
       5C  4937       osp$set_status_condition (r1_status.condition, status);
       70  4938     ELSE
       70  4939       pointer.cell_pointer := NIL;
       78  4940     IFEND;
       78  4941
       78  4942   PROCEND mmp$close_shared_stack;
```

MMP$CREATE_SCRATCH_SEGMENT

```
o  4944 {
o  4945 {   The purpose of this request is to create a scratch segment.
o  4946 {   Scratch segments are temporary segments that have no name and
o  4947 {   exist only until deleted via the mmp$delete_scratch_segment request,
o  4948 {   or until the creating task terminates.
o  4949 {
o  4950 {       MMP$CREATE_SCRATCH_SEGMENT (POINTER_KIND, ACCESS_SELECTIONS,
o  4951 {             POINTER, STATUS)
o  4952 {
o  4953 { POINTER_KIND: (input) This parameter specifies the type of pointer
o  4954 {       to be constructed for the segment.
o  4955 {
o  4956 { ACCESS_SELECTIONS: (input) This parameter specifies the mode of
o  4957 {       access to the segment (sequential or random).
o  4958 {
o  4959 { POINTER: (output) This parameter specifies the process virtual
o  4960 {       address assigned to the segment.  The byte offset in the PVA
o  4961 {       is set to zero.
o  4962 {
o  4963 { STATUS: (output) This parameter specifies the request status.
o  4964 {       The possible error codes are:
o  4965 {             dme$unable_to_locate_fde
o  4966 {             dme$unable_to_get_fd_lock
o  4967 {             mme$address_not_0_mod_16384
o  4968 {             mme$asid_specified
o  4969 {             mme$binding_attribute_invalid
o  4970 {             mme$contig_mem_seg_violation
o  4971 {             mme$execute_global_invalid
o  4972 {             mme$invalid_asid_specified
o  4973 {             mme$invalid_length_requested
o  4974 {             mme$invalid_pva
o  4975 {             mme$invalid_ring_brackets
o  4976 {             mme$invalid_shadow_segment
o  4977 {             mme$invalid_shared_taskid
o  4978 {             mme$length_not_0_mod_16384
o  4979 {             mme$pages_already_assigned
o  4980 {             mme$ref_to_unrecovered_file
o  4981 {             mme$ring_violation
o  4982 {             mme$segment_number_is_in_use
o  4983 {             mme$segment_number_not_in_use
o  4984 {             mme$segment_number_too_big
o  4985 {             mme$segment_origin_change
o  4986 {             mme$segment_origin_invalid
o  4987 {             mme$segment_table_is_full
o  4988 {             mme$software_attribute_invalid
o  4989 {             mme$unable_to_assign_contig_mem
o  4990 {             mme$unsupported_keyword
o  4991 {
o  4992
o  4993
o  4994   PROCEDURE [XDCL, #GATE] mmp$create_scratch_segment
o  4995     (     pointer_kind: amt$pointer_kind;
o  4996           access_selections: mmt$access_selections;
o  4997       VAR pointer: amt$segment_pointer;
o  4998       VAR status: ost$status);
o  4999
```


MMP$CREATE_SCRATCH_SEGMENT

```
o  5000     VAR
o  5001       caller_id: ost$caller_identifier,
o  5002       kind: mmt$segment_pointer_kind,
o  5003       p: mmt$segment_pointer,
o  5004       r1_status: ost$status,
o  5005       segment_attributes: mmt$segment_attrib_descriptor;
o  5006
o  5007     #CALLER_ID (caller_id);
C  5008     #KEYPOINT (osk$entry, 0, mmk$create_scratch_segment);
18  5009     status.normal := TRUE;
18  5010     IF pointer_kind = amc$cell_pointer THEN
2A  5011       kind := mmc$cell_pointer;
30  5012     ELSEIF pointer_kind = amc$sequence_pointer THEN
36  5013       kind := mmc$sequence_pointer;
3E  5014     ELSE
3E  5015       kind := mmc$heap_pointer;
42  5016     IFEND;
42  5017     segment_attributes.validating_ring_number := caller_id.ring;
42  5018     segment_attributes.file_limits_to_enforce := sfc$temp_file_space_limit;
42  5019     segment_attributes.pointer_kind := kind;
42  5020     segment_attributes.user_attributes := NIL;
42  5021     segment_attributes.sfid := gfv$null_sfid;
42  5022     mmp$build_segment (segment_attributes, NIL {shared_taskid_array} , p, r1_status);
98  5023
98  5024     IF r1_status.normal THEN
A0  5025       IF access_selections = mmc$as_sequential THEN
A8  5026         mmp$set_access_selections (p.cell_pointer, mmc$as_sequential, status);
C4  5027       IFEND;
C4  5028       pointer.kind := pointer_kind;
C4  5029       IF pointer_kind = amc$cell_pointer THEN
D4  5030         pointer.cell_pointer := p.cell_pointer;
E0  5031       ELSEIF pointer_kind = amc$sequence_pointer THEN
E4  5032         pointer.sequence_pointer := p.seq_pointer;
F2  5033       ELSE
F2  5034         pointer.heap_pointer := p.heap_pointer;
FC  5035       IFEND;
100  5036     ELSE
100  5037       osp$set_status_condition (r1_status.condition, status);
112  5038     IFEND;
112  5039     #KEYPOINT (osk$exit, 0, mmk$create_scratch_segment);
116  5040
116  5041   PROCEND mmp$create_scratch_segment;
  0  5042
```

```
o  5044 {
o  5045 {   The purpose of this request is to create a transient segment.
o  5046 {
o  5047 {       MMP$CREATE_SEGMENT (SEG_ATTRIBUTES_P, POINTER_KIND,
o  5048 {         VALIDATION_RING_NUMBER, POINTER, STATUS)
o  5049 {
o  5050 { SEG_ATTRIBUTES_P: (input) This parameter is a pointer to the array of segment
o  5051 {       attributes to be assigned to the segment.
o  5052 {
o  5053 { POINTER_KIND: (input) This parameter specifies the type of pointer
o  5054 {       to be constructed for the segment.
o  5055 {
o  5056 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring of
o  5057 {       execution for whom the segment is being created.  The ring
o  5058 {       number used for validation is the max of caller ring number
o  5059 {       and 'validation_ring_number'.
o  5060 {
o  5061 { POINTER: (output) This parameter specifies the process virtual
o  5062 {       address assigned to the segment.  The byte offset in the PVA
o  5063 {       is set to zero
o  5064 {
o  5065 { STATUS: (output) This parameter specifies the request status.
o  5066 {       The possible error codes are:
o  5067 {             dme$unable_to_locate_fde
o  5068 {             dme$unable_to_get_fd_lock
o  5069 {             mme$address_not_0_mod_16384
o  5070 {             mme$asid_specified
o  5071 {             mme$binding_attribute_invalid
o  5072 {             mme$contig_mem_seg_violation
o  5073 {             mme$execute_global_invalid
o  5074 {             mme$invalid_asid_specified
o  5075 {             mme$invalid_length_requested
o  5076 {             mme$invalid_pva
o  5077 {             mme$invalid_ring_brackets
o  5078 {             mme$invalid_shadow_segment
o  5079 {             mme$invalid_shared_taskid
o  5080 {             mme$length_not_0_mod_16384
o  5081 {             mme$pages_already_assigned
o  5082 {             mme$ref_to_unrecovered_file
o  5083 {             mme$ring_violation
o  5084 {             mme$segment_number_is_in_use
o  5085 {             mme$segment_number_not_in_use
o  5086 {             mme$segment_number_too_big
o  5087 {             mme$segment_origin_invalid
o  5088 {             mme$segment_table_is_full
o  5089 {             mme$software_attribute_invalid
o  5090 {             mme$unable_to_assign_contig_mem
o  5091 {             mme$unsupported_keyword
o  5092 {
o  5093
o  5094
o  5095    PROCEDURE [XDCL, #GATE] mmp$create_segment
o  5096      (     seg_attributes_p: ^array [ * ] of mmt$attribute_descriptor;
o  5097            pointer_kind: mmt$segment_pointer_kind;
o  5098            validation_ring_number: ost$valid_ring;
o  5099       VAR pointer: mmt$segment_pointer;
```



```
o  5100       VAR status: ost$status);
o  5101
o  5102
o  5103      VAR
o  5104        caller_id: ost$caller_identifier,
o  5105        r1_pointer: mmt$segment_pointer,
o  5106        r1_status: ost$status,
o  5107        segment_attributes: mmt$segment_attrib_descriptor,
o  5108        validating_ring_number: ost$valid_ring;
o  5109
o  5110      #CALLER_ID (caller_id);
C  5111      #KEYPOINT (osk$entry, 0, mmk$create_segment);
18  5112     status.normal := TRUE;
18  5113     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
3A  5114     segment_attributes.validating_ring_number := validating_ring_number;
3A  5115     segment_attributes.file_limits_to_enforce := sfc$no_limit;
3A  5116     segment_attributes.pointer_kind := pointer_kind;
3A  5117     segment_attributes.user_attributes := seg_attributes_p;
5E  5118     segment_attributes.sfid := gfv$null_sfid;
5E  5119     IF segment_attributes.user_attributes <> NIL THEN
78  5120       PUSH segment_attributes.user_attributes: [LOWERBOUND (seg_attributes_p^
AE  5121             ) .. UPPERBOUND (seg_attributes_p^)];
AE  5122       segment_attributes.user_attributes^ := seg_attributes_p^;
EE  5123     IFEND;
EE  5124     mmp$build_segment (segment_attributes, NIL (shared_taskid_array) , r1_pointer, r1_status);
11E  5125    IF NOT r1_status.normal THEN
126  5126      osp$set_status_condition (r1_status.condition, status);
13A  5127    ELSE
13A  5128      pointer := r1_pointer;
148  5129    IFEND;
148  5130    #KEYPOINT (osk$exit, 0, mmk$create_segment);
14C  5131
14C  5132  PROCEND mmp$create_segment;
```

```
      0  5134 {
      0  5135 {   The purpose of this request is to create a transient segment that uses the
      0  5136 { ACTIVE file of an existing segment as a PASSIVE file.
      0  5137 {
      0  5138 {    MMP$CREATE_SHADOW_SEGMENT (shadow_pva, shadow_offset, shadow_length,
      0  5139 {          pointer_kind, pva, status);
      0  5140 {
      0  5141 { SHADOW_PVA :(input) This parameter specifies the segment pointer of the
      0  5142 {      segment to be shadowed.
      0  5143 {
      0  5144 { SHADOW_OFFSET: (input) This parameter specifies the byte offset within the
      0  5145 {      segment from which to start shadowing.
      0  5146 {
      0  5147 { SHADOW_LENGTH: (input) This parameter specifies the length of portion in
      0  5148 {      segment to be shadowed.
      0  5149 {
      0  5150 { POINTER_KIND: (input) This parameter specifies the type of pointer to be
      0  5151 {      constructed for the segment.
      0  5152 {
      0  5153 { PVA: (output) This parameter returns the process virtual address of the
      0  5154 {      transient segment.
      0  5155 {
      0  5156 { STATUS: (output) This parameter specifies the request status returned.
      0  5157 {
      0  5158
      0  5159
      0  5160
      0  5161   PROCEDURE [XDCL, #GATE] mmp$create_shadow_segment
      0  5162     (    segment_p: ^cell;
      0  5163          shadow_offset: ost$segment_offset;
      0  5164          shadow_length: ost$segment_length;
      0  5165          pointer_kind: mmt$segment_pointer_kind;
      0  5166      VAR pva: mmt$segment_pointer;
      0  5167      VAR status: ost$status);
      0  5168
      0  5169     VAR
      0  5170       caller_id: ost$caller_identifier,
      0  5171       conv_ptr: ^cell,
      0  5172       i: pmt$initialization_value,
      0  5173       preset_value: pmt$initialization_value,
      0  5174       r1_pointer: mmt$segment_pointer,
      0  5175       r1_status: ost$status,
      0  5176       segment_attributes: mmt$segment_attrib_descriptor,
      0  5177       segnum: ost$segment;
      0  5178
      0  5179     #KEYPOINT (osk$entry, #SEGMENT (segment_p) * osk$m, mmk$create_shadow_segment);
     1E  5180     status.normal := TRUE;
     1E  5181
     1E  5182     conv_ptr := #ADDRESS (#RING (segment_p), #SEGMENT (segment_p), shadow_offset);
     1E  5183
     1E  5184 {  Issue request to create ACTIVE segment.
     1E  5185     #CALLER_ID (caller_id);
     1E  5186     segment_attributes.validating_ring_number := caller_id.ring;
     1E  5187     segment_attributes.file_limits_to_enforce := sfc$no_limit;
     1E  5188     segment_attributes.pointer_kind := pointer_kind;
     1E  5189     PUSH segment_attributes.user_attributes: [1 .. 1];
```



```
     72  5190     segment_attributes.user_attributes^ [1].keyword := mmc$kw_shadow_segment;
     72  5191     segment_attributes.user_attributes^ [1].shadow_p := conv_ptr;
     72  5192     segment_attributes.user_attributes^ [1].shadow_length := shadow_length;
     72  5193     segment_attributes.sfid := gfv$null_sfid;
     72  5194     mmp$build_segment (segment_attributes, NIL [shared_taskid_array] , r1_pointer, r1_status);
     E6  5195     IF NOT r1_status.normal THEN
     EE  5196       osp$set_status_condition (r1_status.condition, status);
    102  5197     ELSE
    102  5198       pva := r1_pointer;
    110  5199     IFEND;
    110  5200     #KEYPOINT (osk$exit, 0, mmk$create_shadow_segment);
    114  5201
    114  5202   PROCEND mmp$create_shadow_segment;
      0  5203
```

MMP$CREATE_SHARED_STACK

```
o  5205 {
o  5206 {    The purpose of this request is to create a stack segment to be
o  5207 { used by ADA tasks.
o  5208 {
o  5209 { MMP$CREATE_SHARED_STACK (SEG_ATTRIBUTES_P, POINTER_KIND,
o  5210 {   SHARED_TASKID_ARRAY, POINTER, STATUS)
o  5211 {
o  5212 { SEG_ATTRIBUTES_P: (INPUT) This parameter is a pointer to the array of segment
o  5213 {                             attributes to be assigned to the segment.
o  5214 {
o  5215 { POINTER_KIND: (INPUT) This parameter specifies the type of pointer to
o  5216 {                        be constructed for the segment.
o  5217 {
o  5218 { SHARED_TASKID_ARRAY: (INPUT) This parameter specifies the array of taskids
o  5219 {                                for the tasks in which this segment is being opened.
o  5220 {
o  5221 { POINTER: (OUTPUT) This parameter specifies the process virtual address assigned
o  5222 {                   to the segment. The byte offset in the PVA is set to zero.
o  5223 {
o  5224 { STATUS: (OUTPUT) This parameter specifies the request status.
o  5225 {
o  5226
o  5227   PROCEDURE [XDCL] mmp$create_shared_stack
o  5228     (     seg_attributes_p: ^array [ * ] of mmt$attribute_descriptor;
o  5229           pointer_kind: mmt$segment_pointer_kind;
o  5230           shared_taskid_array: ^array [1 .. * ] of pmt$task_id;
o  5231      VAR pointer: mmt$segment_pointer;
o  5232      VAR status: ost$status);
o  5233
o  5234     VAR
o  5235       caller_id: ost$caller_identifier,
o  5236       r1_pointer: mmt$segment_pointer,
o  5237       r1_status: ost$status,
o  5238       segment_attributes: mmt$segment_attrib_descriptor;
o  5239
o  5240     #CALLER_ID (caller_id);
C  5241     status.normal := TRUE;
C  5242     segment_attributes.validating_ring_number := caller_id.ring;
C  5243     segment_attributes.file_limits_to_enforce := sfc$no_limit;
C  5244     segment_attributes.pointer_kind := pointer_kind;
C  5245     segment_attributes.user_attributes := seg_attributes_p;
42 5246     IF segment_attributes.user_attributes <> NIL THEN
50 5247       PUSH segment_attributes.user_attributes: [LOWERBOUND (seg_attributes_p^
86 5248            ) .. UPPERBOUND (seg_attributes_p^)];
86 5249       segment_attributes.user_attributes^ := seg_attributes_p^;
C6 5250     IFEND;
C6 5251     segment_attributes.sfid := gfv$null_sfid;
C6 5252     mmp$build_segment (segment_attributes, shared_taskid_array, r1_pointer, r1_status);
FE 5253     IF NOT r1_status.normal THEN
106 5254      osp$set_status_condition (r1_status.condition, status);
118 5255    ELSE
118 5256      pointer := r1_pointer;
126 5257    IFEND;
126 5258
126 5259   PROCEND mmp$create_shared_stack;
```


MMP$CREATE_USER_SEGMENT

```
o  5262
o  5263 { The purpose of this request is to create a transient segment.
o  5264 {
o  5265 {  MMP$CREATE_USER_SEGMENT (segment_attributes_p, pointer_kind, access_selections,
o  5266 {           pointer, status)
o  5267 {
o  5268 {   SEGMENT_ATTRIBUTES_P: (INPUT) This parameter is a pointer to an array
o  5269 {           of segment_attributes to be assigned to the segment.
o  5270 {
o  5271 {   POINTER_KIND: (INPUT) This parameter specifies the type of pointer
o  5272 {           to be constructed for the segment.
o  5273 {
o  5274 {   ACCESS_SELECTIONS: (INPUT) This parameter specifies  the mode of access
o  5275 {           to the segment (sequential or random).
o  5276 {
o  5277 {   POINTER: (OUTPUT) This parameter specifies the process virtual address
o  5278 {           assigned to the segment. The byte offset in the PVA is set to zero.
o  5279 {
o  5280 {   STATUS: (OUTPUT) This parameter specifies the request status.
o  5281 {           The possible error codes are:
o  5282 {                dme$unable_to_locate_fde
o  5283 {                dme$unable_to_get_fd_lock
o  5284 {                mme$address_not_0_mod_16384
o  5285 {                mme$asid_specified
o  5286 {                mme$binding_attribute_invalid
o  5287 {                mme$contig_mem_seg_violation
o  5288 {                mme$execute_global_invalid
o  5289 {                mme$invalid_asid_specified
o  5290 {                mme$invalid_length_requested
o  5291 {                mme$invalid_pva
o  5292 {                mme$invalid_ring_brackets
o  5293 {                mme$invalid_shadow_segment
o  5294 {                mme$invalid_shared_taskid
o  5295 {                mme$length_not_0_mod_16384
o  5296 {                mme$pages_already_assigned
o  5297 {                mme$ref_to_unrecovered_file
o  5298 {                mme$ring_violation
o  5299 {                mme$segment_number_is_in_use
o  5300 {                mme$segment_number_not_in_use
o  5301 {                mme$segment_number_too_big
o  5302 {                mme$segment_origin_change
o  5303 {                mme$segment_origin_invalid
o  5304 {                mme$segment_table_is_full
o  5305 {                mme$software_attribute_invalid
o  5306 {                mme$unable_to_assign_contig_mem
o  5307 {                mme$unsupported_keyword
o  5308 {                mme$wired_seg_length_too_large
o  5309 {
o  5310
o  5311
o  5312   PROCEDURE [XDCL, #GATE] mmp$create_user_segment
o  5313     (     user_attributes_p: ^array [ * ] of mmt$user_attribute_descriptor;
o  5314           pointer_kind: amt$pointer_kind;
o  5315           access_selections: mmt$access_selections;
o  5316      VAR pointer: amt$segment_pointer;
o  5317      VAR status: ost$status);
```

```
      0  5318
      0  5319      VAR
      0  5320        byte: 0 .. 255,
      0  5321        caller_id: ost$caller_identifier,
      0  5322        contiguous_flag: boolean,
      0  5323        contiguous_page_count: integer,
      0  5324        file_limits_to_enforce: sft$file_space_limit_kind,
      0  5325        i: integer,
      0  5326        increment: integer,
      0  5327        kind: mmt$segment_pointer_kind,
      0  5328        local_status: ost$status,
      0  5329        max_length_index: integer,
      0  5330        max_length_specified: boolean,
      0  5331        page_size: integer,
      0  5332        preset_pointer: ^array [ * ] of 0 .. 255,
      0  5333        r1_status: ost$status,
      0  5334        save_index: integer,
      0  5335        seg_attrib_p: ^array [ * ] of mmt$attribute_descriptor,
      0  5336        segment_attributes: mmt$segment_attrib_descriptor,
      0  5337        segment_pointer: mmt$segment_pointer,
      0  5338        seq_p: ^SEQ ( * ),
      0  5339        wired_flag: boolean,
      0  5340        wired_index: integer;
      0  5341
      0  5342      contiguous_flag := FALSE;
      C  5343      status.normal := TRUE;
      C  5344      max_length_specified := FALSE;
      C  5345      wired_flag := FALSE;
      C  5346
      C  5347      IF user_attributes_p <> NIL THEN
     30  5348        PUSH seg_attrib_p: [LOWERBOUND (user_attributes_p^) .. UPPERBOUND (user_attributes_p^)];
     62  5349
     62  5350        FOR i := LOWERBOUND (user_attributes_p^) TO UPPERBOUND (user_attributes_p^) DO
     70  5351          CASE user_attributes_p^ [i].keyword OF
     BC  5352          = mmc$ua_ring_numbers =
     BC  5353            seg_attrib_p^ [i].keyword := mmc$kw_ring_numbers;
     BC  5354            seg_attrib_p^ [i].r1 := user_attributes_p^ [i].r1;
     BC  5355            seg_attrib_p^ [i].r2 := user_attributes_p^ [i].r2;
     FC  5356          = mmc$ua_segment_number =
     FC  5357            seg_attrib_p^ [i].keyword := mmc$kw_segment_number;
     FC  5358            seg_attrib_p^ [i].segnum := user_attributes_p^ [i].segnum;
    11E  5359          = mmc$ua_max_segment_length =
    11E  5360            seg_attrib_p^ [i].keyword := mmc$kw_max_segment_length;
    11E  5361            seg_attrib_p^ [i].max_length := user_attributes_p^ [i].max_length;
    11E  5362            max_length_specified := TRUE;
    11E  5363            max_length_index := i;
    14C  5364          = mmc$ua_preset_value =
    14C  5365            seg_attrib_p^ [i].keyword := mmc$kw_preset_value;
    14C  5366            seg_attrib_p^ [i].preset_value := user_attributes_p^ [i].preset_value;
    170  5367          = mmc$ua_segment_access_control =
    170  5368            seg_attrib_p^ [i].keyword := mmc$kw_segment_access_control;
    170  5369            seg_attrib_p^ [i].access_control := user_attributes_p^ [i].access_control;
    1AA  5370          = mmc$ua_wired_segment =
    1AA  5371            seg_attrib_p^ [i].keyword := mmc$kw_wired_segment;
    1AA  5372            IF user_attributes_p^ [i].wired_segment_length > 65536 THEN
    1E4  5373              osp$set_status_abnormal ('MM', mme$wired_seg_length_too_large, '', status);
```



```
    214  5374              RETURN;
    216  5375            IFEND;
    216  5376            seg_attrib_p^ [i].wired_segment_length := user_attributes_p^ [i].wired_segment_length;
    216  5377            IF user_attributes_p^ [i].contiguous_real_memory THEN
    24E  5378              contiguous_flag := TRUE;
    252  5379            IFEND;
    252  5380            wired_flag := TRUE;
    252  5381            wired_index := i;
    260  5382          = mmc$ua_null_keyword =
    260  5383            seg_attrib_p^ [i].keyword := mmc$kw_null_keyword;
    27A  5384          ELSE
    27A  5385          CASEND;
    27A  5386        FOREND;
    286  5387      ELSE
    286  5388        seg_attrib_p := NIL;
    28A  5389      IFEND;
    28A  5390
    28A  5391      IF max_length_specified AND wired_flag THEN
    292  5392        seg_attrib_p^ [max_length_index].max_length := seg_attrib_p^ [wired_index].wired_segment_length;
    2C0  5393      IFEND;
    2C0  5394
    2C0  5395      IF pointer_kind = amc$cell_pointer THEN
    2C8  5396        kind := mmc$cell_pointer;
    2CE  5397      ELSEIF pointer_kind = amc$sequence_pointer THEN
    2D4  5398        kind := mmc$sequence_pointer;
    2DC  5399      ELSE
    2DC  5400        kind := mmc$heap_pointer;
    2E0  5401      IFEND;
    2E0  5402      #CALLER_ID (caller_id);
    2E0  5403
    2E0  5404      segment_attributes.validating_ring_number := caller_id.ring;
    2E0  5405      segment_attributes.file_limits_to_enforce := sfc$temp_file_space_limit;
    2E0  5406      segment_attributes.pointer_kind := kind;
    2E0  5407      segment_attributes.user_attributes := seg_attrib_p;
    304  5408      segment_attributes.sfid := gfv$null_sfid;
    304  5409      mmp$build_segment (segment_attributes, NIL {shared_taskid_array} , segment_pointer, r1_status);
    340  5410
    340  5411      IF r1_status.normal THEN
    348  5412        IF access_selections = mmc$as_sequential THEN
    350  5413          mmp$set_access_selections (segment_pointer.cell_pointer, mmc$as_sequential, status);
    36C  5414        IFEND;
    36C  5415        pointer.kind := pointer_kind;
    36C  5416        IF pointer_kind = amc$cell_pointer THEN
    37C  5417          seq_p := #SEQ (segment_pointer.cell_pointer^);
    37C  5418          RESET seq_p;
    37C  5419          NEXT preset_pointer: [1 .. #SIZE (segment_pointer.cell_pointer^)] IN seq_p;
    3CA  5420          pointer.cell_pointer := segment_pointer.cell_pointer;
    3D6  5421        ELSEIF pointer_kind = amc$sequence_pointer THEN
    3DA  5422          seq_p := #SEQ (segment_pointer.seq_pointer^);
    3DA  5423          RESET seq_p;
    3DA  5424          NEXT preset_pointer: [1 .. #SIZE (segment_pointer.seq_pointer^)] IN seq_p;
    42C  5425          pointer.sequence_pointer := segment_pointer.seq_pointer;
    43A  5426        ELSE
    43A  5427          seq_p := #SEQ (segment_pointer.heap_pointer^);
    43A  5428          RESET seq_p;
    43A  5429          NEXT preset_pointer: [1 .. #SIZE (segment_pointer.heap_pointer^)] IN seq_p;
```

MMP$CREATE_USER_SEGMENT

```
48C  5430          pointer.heap_pointer := segment_pointer.heap_pointer;
496  5431        IFEND;
496  5432
496  5433        IF contiguous_flag THEN
49A  5434          mmp$assign_contiguous_memory (pointer.cell_pointer, seg_attrib_p^ [wired_index].wired_segment_length,
4DO  5435                status);
4DO  5436
4DO  5437          { MMP$ASSIGN_CONTIGUOUS_MEMORY will PRESET the pages it assigns to the segment--
4DO  5438          { if it was able to assign contiguous pages.
4DO  5439
4DO  5440          IF NOT status.normal THEN
4D8  5441            mmp$delete_user_segment (pointer, local_status);
4FO  5442            RETURN;
4F2  5443          IFEND;
4F6  5444        ELSEIF wired_flag AND NOT contiguous_flag THEN
4FE  5445
4FE  5446          { PRESET the pages of a wired segment-this is accomplished by touching every page in the segment.}
4FE  5447
4FE  5448          mmp$get_page_size (page_size);
504  5449          increment := 1;
504  5450          WHILE increment < UPPERBOUND (preset_pointer^) DO
528  5451            byte := preset_pointer^ [increment];
528  5452            increment := increment + page_size;
528  5453          WHILEND;
554  5454        IFEND;
556  5455      ELSE
556  5456        osp$set_status_condition (r1_status.condition, status);
566  5457      IFEND;
566  5458
566  5459    PROCEND mmp$create_user_segment;
```


MMP$DELETE_SCRATCH_SEGMENT

```
 O  5461 {
 O  5462 {   The purpose of this request is to delete a scratch segment.
 O  5463 {
 O  5464 {       MMP$DELETE_SCRATCH_SEGMENT (POINTER, STATUS)
 O  5465 {
 O  5466 { POINTER: (input_output) This parameter specifies the segment
 O  5467 {       to be deleted.  This pointer will be set to NIL when the request
 O  5468 {       completes.
 O  5469 {
 O  5470 { STATUS: (output) This parameter specifies the request status.
 O  5471 {       The possible error codes are:
 O  5472 {             dme$file_descriptor_not_deleted
 O  5473 {             mme$invalid_close_segment_req
 O  5474 {             mme$invalid_shared_taskid
 O  5475 {             mme$segment_number_not_in_use
 O  5476 {             mme$segment_number_too_big
 O  5477 {
 O  5478
 O  5479   PROCEDURE [XDCL, #GATE] mmp$delete_scratch_segment
 O  5480     (VAR pointer: amt$segment_pointer;
 O  5481      VAR status: ost$status);
 O  5482
 O  5483     VAR
 O  5484       caller_id: ost$caller_identifier,
 O  5485       p: mmt$segment_pointer,
 O  5486       r1_status: ost$status;
 O  5487
 O  5488     #CALLER_ID (caller_id);
  C  5489     #KEYPOINT (osk$entry, #SEGMENT (pointer.cell_pointer) * osk$m, mmk$delete_scratch_segment);
 2A  5490     status.normal := TRUE;
 2A  5491     p.kind := mmc$cell_pointer;
 2A  5492     p.cell_pointer := pointer.cell_pointer;
 2A  5493
 2A  5494     mmp$invalidate_segment (#SEGMENT (pointer.cell_pointer), caller_id.ring, NIL (shared_taskid_array) ,
 78  5495           r1_status);
 78  5496     IF r1_status.normal THEN
 80  5497       pointer.cell_pointer := NIL;
 88  5498     ELSE
 88  5499       osp$set_status_condition (r1_status.condition, status);
 98  5500     IFEND;
 98  5501
 98  5502     #KEYPOINT (osk$exit, 0, mmk$delete_scratch_segment);
 9C  5503
 9C  5504   PROCEND mmp$delete_scratch_segment;
  O  5505
```

MMP$DELETE_SEGMENT

```
0  5508 {
0  5509 {   The purpose of this request is to delete/close a segment.  If the
0  5510 { caller is not within the read bracket of the segment being deleted the
0  5511 { call is rejected.
0  5512 {
0  5513 {       MMP$DELETE_SEGMENT (POINTER, VALIDATION_RING_NUMBER, STATUS)
0  5514 {
0  5515 { POINTER: (input_output) This parameter specifies the segment
0  5516 {       to be deleted.  This pointer will be set to NIL when the request
0  5517 {       completes.
0  5518 {
0  5519 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring of
0  5520 {       execution for whom the segment is being deleted.  The ring
0  5521 {       number used for validation is the max of caller ring number
0  5522 {       and 'validation_ring_number'.
0  5523 {
0  5524 { STATUS: (output) This parameter specifies the request status.
0  5525 {       The possible error codes are:
0  5526 {             dme$file_descriptor_not_deleted
0  5527 {             mme$invalid_close_segment_req
0  5528 {             mme$invalid_shared_taskid
0  5529 {             mme$segment_number_not_in_use
0  5530 {             mme$segment_number_too_big
0  5531 {
0  5532
0  5533
0  5534
0  5535   PROCEDURE [XDCL, #GATE] mmp$delete_segment
0  5536     (VAR pointer: mmt$segment_pointer;
0  5537          validation_ring_number: ost$valid_ring;
0  5538      VAR status: ost$status);
0  5539
0  5540     VAR
0  5541       caller_id: ost$caller_identifier,
0  5542       r1_status: ost$status,
0  5543       validating_ring_number: ost$valid_ring;
0  5544
0  5545     #CALLER_ID (caller_id);
 C 5546     #KEYPOINT (osk$entry, #SEGMENT (pointer.cell_pointer) * osk$m, mmk$delete_segment);
2A 5547     status.normal := TRUE;
2A 5548     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
4C 5549     mmp$invalidate_segment (#SEGMENT (pointer.cell_pointer), validating_ring_number,
86 5550           NIL (shared_taskid_array), r1_status);
86 5551     IF r1_status.normal THEN
8E 5552       pointer.cell_pointer := NIL;
96 5553     ELSE
96 5554       osp$set_status_condition (r1_status.condition, status);
A8 5555     IFEND;
A8 5556     #KEYPOINT (osk$exit, 0, mmk$delete_segment);
AC 5557
AC 5558   PROCEND mmp$delete_segment;
```


MMP$DELETE_USER_SEGMENT

```
0  5561
0  5562 {
0  5563 { The purpose of this request is to close/delete a segment.
0  5564 {
0  5565 {  MMP$DELETE_USER_SEGMENT (pointer, status)
0  5566 {
0  5567 {  POINTER: (INPUT/OUTPUT) This parameter specifies the segment
0  5568 {         to be deleted. This pointer will be set to NIL when the
0  5569 {         request completes.
0  5570 {
0  5571 {  STATUS: (OUTPUT) This parameter specifies the request status.
0  5572 {         The possible error codes are:
0  5573 {               dme$file_descriptor_not_deleted
0  5574 {               mme$invalid_close_segment_req
0  5575 {               mme$invalid_shared_taskid
0  5576 {               mme$segment_number_not_in_use
0  5577 {               mme$segment_number_too_big
0  5578 {
0  5579
0  5580
0  5581   PROCEDURE [XDCL, #GATE] mmp$delete_user_segment
0  5582     (VAR pointer: amt$segment_pointer;
0  5583      VAR status: ost$status);
0  5584
0  5585     VAR
0  5586       caller_id: ost$caller_identifier,
0  5587       r1_status: ost$status;
0  5588
0  5589     #CALLER_ID (caller_id);
 C 5590     status.normal := TRUE;
 C 5591     mmp$invalidate_segment (#SEGMENT (pointer.cell_pointer), caller_id.ring, NIL (shared_taskid_array) ,
5C 5592           r1_status);
5C 5593     IF r1_status.normal THEN
64 5594       pointer.cell_pointer := NIL;
6A 5595     ELSE
6A 5596       osp$set_status_condition (r1_status.condition, status);
7C 5597     IFEND;
7C 5598
7C 5599   PROCEND mmp$delete_user_segment;
```

```
 0  5602
 0  5603 {  The purpose of the procedure is to attempt to complete file
 0  5604 {allocation on a file which has previously failed in an
 0  5605 {attempt to allocate file space. It only handles cases
 0  5606 {where the failure was dme$unable_to_alloc_all_space.
 0  5607 {  The ring 1 segment manager sets a system flag if
 0  5608 {such a failure occurs.
 0  5609 {  The procedure make a single attempt to perform
 0  5610 {file allocation. If allocation is not possible then
 0  5611 {the procedure will wait a short time and then return
 0  5612 {to the code which caused the allocation to be required.
 0  5613 {If any events such as terminate_job or interactive break have
 0  5614 {arisen during the wait, they will be processed on the return
 0  5615 {if possible.  If nothing preempts the original code, it will cause
 0  5616 {the same allocation requirement and the process is repeated.
 0  5617
 0  5618   PROCEDURE [XDCL] mmp$failed_allocation_flag_hdl
 0  5619     (    flag_id: ost$system_flag);
 0  5620
 0  5621     VAR
 0  5622       i: integer,
 0  5623       new_allocated_length: amt$file_byte_address,
 0  5624       previous_allocated_length: amt$file_byte_address,
 0  5625       segnum: ost$segment,
 0  5626       status: ost$status,
 0  5627       wait_time: integer,
 0  5628       xcb_p: ^ost$execution_control_block;
 0  5629
 0  5630     pmp$find_executing_task_xcb (xcb_p);
 A  5631     previous_allocated_length := 0;
 A  5632
 A  5633     WHILE TRUE DO
26  5634       IF pmp$task_state () = pmc$task_terminating THEN
38  5635         RETURN;
3A  5636       IFEND;
3A  5637
3A  5638       wait_time := mmv$file_allocation_interval;
3A  5639       mmp$process_file_alloc (new_allocated_length, status);
5E  5640       IF status.normal THEN
66  5641         RETURN;
6C  5642       ELSEIF NOT (status.normal) AND (status.condition = dme$unable_to_alloc_all_space) THEN
7C  5643         IF (new_allocated_length > 1000000) OR (new_allocated_length > previous_allocated_length) THEN
8C  5644           wait_time := 2000;
90  5645         IFEND;
90  5646         previous_allocated_length := new_allocated_length;
94  5647       IFEND;
94  5648
94  5649 { Any abnormal status which is actually returned to this procedure is an 'OK' status, which
94  5650 { will cause the system to keep attempting to allocate file_space for this task.
94  5651 { In the case of the File_Server, however, we don't want to wait forever because the server
94  5652 { may not EVER come back.  Just return; the next time the user references the page the correct
94  5653 { condition will be raised.
94  5654
94  5655       IF (status.condition = dfe$family_not_served) OR (status.condition = dfe$server_not_active) OR
B0  5656           (status.condition = dfe$server_has_terminated) THEN
B0  5657         RETURN;
```



```
B2  5658       IFEND;
B2  5659
B2  5660       bap$exit_fap_on_condition (dme$unable_to_alloc_all_space);
C6  5661
C6  5662       {Special trap for system disk full in early deadstart
C6  5663       IF NOT dsp$system_committed () THEN
D6  5664         dpp$put_critical_message ('The system disk is full - redeadstart without job recovery', status);
FA  5665       IFEND;
FA  5666
FA  5667       {Raise the appropriate user condition.
FA  5668       osp$wait_on_condition (dme$unable_to_alloc_all_space);
10E 5669
10E 5670       IF (xcb_p^.system_flags * $tmt$system_flags [pmc$sf_terminate_task, jmc$terminate_job_flag,
12C 5671             jmc$logout_flag_id, jmc$kill_job_flag, pmc$kill_task_flag] <> $tmt$system_flags []) AND
12C 5672             (xcb_p^.system_table_lock_count <= 0) THEN
12C 5673         pmp$log (' Job terminated while waiting for file allocation.', status);
150 5674         pmp$exit (status);
17C 5675       IFEND;
17C 5676
17C 5677       IF (xcb_p^.system_flags = $tmt$system_flags []) THEN
188 5678         FOR i := 1 TO tmc$maximum_signals DO
190 5679           IF xcb_p^.signals.present [i] THEN
1A4 5680             RETURN;
1A6 5681           IFEND;
1A6 5682         FOREND;
1B0 5683       ELSE
1B0 5684         RETURN;
1B2 5685       IFEND;
1B2 5686     WHILEND;
1B6 5687
1B6 5688   PROCEND mmp$failed_allocation_flag_hdl;
```

MMP$FETCH_OFFSET_MODIFIED_PAGES

```
        o  5690
        o  5691    PROCEDURE [XDCL, #GATE] mmp$fetch_offset_modified_pages
        o  5692      (     segment_p: ^cell;
        o  5693            return_unallocated_offsets: boolean;
        o  5694       VAR offset_list: array [ * ] of ost$segment_offset;
        o  5695       VAR offsets_returned: integer;
        o  5696       VAR status: ost$status);
        o  5697
        o  5698      VAR
        o  5699        i: integer,
        o  5700        r1_offset_list_p: ^array [ * ] of ost$segment_offset,
        o  5701        r1_offsets_returned: integer,
        o  5702        r1_status: ost$status;
        o  5703
        o  5704      status.normal := TRUE;
        4  5705      PUSH r1_offset_list_p: [1 .. UPPERBOUND (offset_list)];
       40  5706      mmp$fetch_offset_mod_pages_r1 (#SEGMENT (segment_p), gfv$null_sfid, return_unallocated_offsets,
       8A  5707          r1_offset_list_p, r1_offsets_returned, r1_status);
       8A  5708      IF NOT r1_status.normal THEN
       92  5709        osp$set_status_condition (r1_status.condition, status);
       A6  5710      ELSE
       A6  5711        offset_list := r1_offset_list_p^;
       F2  5712        offsets_returned := r1_offsets_returned;
       FE  5713      IFEND;
       FE  5714
       FE  5715    PROCEND mmp$fetch_offset_modified_pages;
        0  5716
```


MMP$FETCH_SEGMENT_ATTRIBUTES

```
        o  5719 {
        o  5720 {   The purpose of this request is to fetch one or more attributes
        o  5721 { of a segment.
        o  5722 {
        o  5723 {       MMP$FETCH_SEGMENT_ATTRIBUTES (PVA, SEG_ATTRIBUTES, STATUS)
        o  5724 {
        o  5725 { PVA: (input) This parameter specifies the segment number.
        o  5726 {
        o  5727 { SEG_ATTRIBUTES: (input_output) This parameter is an adaptable
        o  5728 {       array of segment attribute descriptors.  Prior to issuing this
        o  5729 {       request, the caller stores into this array the attribute keywords
        o  5730 {       of the attributes to be fetched.
        o  5731 {
        o  5732 { STATUS: (output) This parameter specifies the request status.
        o  5733 {       The possible error codes are:
        o  5734 {             mme$caller_not_in_read_bracket
        o  5735 {             mme$segment_number_not_in_use
        o  5736 {             mme$segment_number_too_big
        o  5737 {             mme$unsupported_keyword
        o  5738 {
        o  5739
        o  5740    PROCEDURE [XDCL, #GATE] mmp$fetch_segment_attributes
        o  5741      (     pva: ^cell;
        o  5742       VAR seg_attributes: array [ * ] of mmt$attribute_descriptor;
        o  5743       VAR status: ost$status);
        o  5744
        o  5745      VAR
        o  5746        caller_id: ost$caller_identifier,
        o  5747        i: integer,
        o  5748        r1_status: ost$status,
        o  5749        r1_segment_attributes_p: ^array [ * ] of mmt$attribute_descriptor;
        o  5750
        o  5751      #KEYPOINT (osk$entry, osk$m * #SEGMENT (pva), mmk$fetch_seg_attributes);
       1E  5752
       1E  5753      status.normal := TRUE;
       1E  5754      #CALLER_ID (caller_id);
       1E  5755      PUSH r1_segment_attributes_p: [LOWERBOUND (seg_attributes) .. UPPERBOUND (seg_attributes)];
       60  5756      r1_segment_attributes_p^ := seg_attributes;
       AC  5757      mmp$fetch_segment_attributes_r1 (#SEGMENT (pva), caller_id.ring, r1_segment_attributes_p, r1_status);
       DE  5758      IF NOT r1_status.normal THEN
       E6  5759        osp$set_status_condition (r1_status.condition, status);
       FC  5760      ELSE
       FC  5761        seg_attributes := r1_segment_attributes_p^;
      140  5762      IFEND;
      140  5763
      140  5764      #KEYPOINT (osk$exit, 0, mmk$fetch_seg_attributes);
      144  5765    PROCEND mmp$fetch_segment_attributes;
```

MMP$GET_ALLOCATED_ADDRESSES

```
        o  5767
        o  5768    PROCEDURE [XDCL, #GATE] mmp$get_allocated_addresses
        o  5769      (     file: ^cell;
        o  5770            starting_byte_address: ost$segment_offset;
        o  5771       VAR addr_list: array [ * ] of dmt$addr_length_pair;
        o  5772       VAR addr_returned: integer;
        o  5773       VAR list_overflow: boolean;
        o  5774       VAR status: ost$status);
        o  5775
        o  5776      VAR
        o  5777        caller_id: ost$caller_identifier,
        o  5778        i: integer,
        o  5779        r1_addr_list_p: ^array [ * ] of dmt$addr_length_pair,
        o  5780        r1_addr_returned: integer,
        o  5781        r1_list_overflow: boolean,
        o  5782        r1_status: ost$status;
        o  5783
        o  5784      status.normal := TRUE;
        C  5785      #CALLER_ID (caller_id);
        C  5786      PUSH r1_addr_list_p: [LOWERBOUND (addr_list) .. UPPERBOUND (addr_list)];
       54  5787      mmp$get_allocated_addresses_r1 (#SEGMENT (file), caller_id.ring, starting_byte_address, r1_addr_list_p,
       A2  5788           r1_addr_returned, r1_list_overflow, r1_status);
       A2  5789      IF NOT r1_status.normal THEN
       AA  5790        osp$set_status_condition (r1_status.condition, status);
       BE  5791      ELSE
       BE  5792        addr_list := r1_addr_list_p^;
      10A  5793        list_overflow := r1_list_overflow;
      10A  5794        addr_returned := r1_addr_returned;
      122  5795      IFEND;
      122  5796    PROCEND mmp$get_allocated_addresses;
        0  5797
```


MMP$GET_SEGMENT_LENGTH

```
        o  5800 {
        o  5801 {   The purpose of this procedure is to return the current segment length
        o  5802 { to the caller.
        o  5803 {
        o  5804 {         MMP$GET_SEGMENT_LENGTH (PVA, VALIDATION_RING_NUMBER, SEGMENT_LENGTH,
        o  5805 {           STATUS)
        o  5806 {
        o  5807 { PVA: (input) This parameter specifies the segment for which the current
        o  5808 {         segment length is returned.
        o  5809 {
        o  5810 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring number
        o  5811 {         used for validation if it is greater than the caller ring number.
        o  5812 {
        o  5813 { SEGMENT_LENGTH: (output) This parameter is where the current segment
        o  5814 {         length is returned.
        o  5815 {
        o  5816 { STATUS: (output) This parameter is where the request status is returned
        o  5817 {         to the caller.  The possible error codes are:
        o  5818 {               mme$caller_not_in_read_bracket
        o  5819 {               mme$invalid_pva
        o  5820 {               mme$ref_to_unrecovered_file
        o  5821 {               mme$segment_number_not_in_use
        o  5822 {               mme$segment_number_too_big
        o  5823 {
        o  5824
        o  5825
        o  5826    PROCEDURE [XDCL, #GATE] mmp$get_segment_length
        o  5827      (     pva: ^cell;
        o  5828            validation_ring_number: ost$valid_ring;
        o  5829       VAR segment_length: ost$segment_length;
        o  5830       VAR status: ost$status);
        o  5831
        o  5832      VAR
        o  5833        caller_id: ost$caller_identifier,
        o  5834        r1_segment_length: ost$segment_length,
        o  5835        r1_status: ost$status,
        o  5836        validating_ring_number: ost$valid_ring;
        o  5837
        o  5838      #KEYPOINT (osk$entry, osk$m * #SEGMENT (pva), mmk$get_segment_length);
       1E  5839      status.normal := TRUE;
       1E  5840      segment_length := 0;
       1E  5841      #CALLER_ID (caller_id);
       1E  5842      determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
       52  5843      mmp$get_segment_length_r1 (#SEGMENT (pva), validating_ring_number, r1_segment_length, r1_status);
       84  5844      IF NOT r1_status.normal THEN
       8C  5845        osp$set_status_condition (r1_status.condition, status);
       A0  5846      ELSE
       A0  5847        segment_length := r1_segment_length;
       A8  5848      IFEND;
       A8  5849      #KEYPOINT (osk$exit, 0, mmk$get_segment_length);
       AC  5850
       AC  5851    PROCEND mmp$get_segment_length;
        0  5852
```

MMP$INITIATE_DEBUG_SHADOWING

```
 0  5855
 0  5856  { The purpose of this request is to initiate shadowing on segments for the
 0  5857  { Interactive Debugger.
 0  5858  {
 0  5859  {  MMP$INITIATE_DEBUG_SHADOWING (segment_pointer, status)
 0  5860  {
 0  5861  { SEGMENT_POINTER: (INPUT) This parameter specifies the process virtual
 0  5862  {         address assigned to the segment.
 0  5863  {
 0  5864  { STATUS: (OUTPUT) This parameter specifies the request status.
 0  5865  {
 0  5866
 0  5867    PROCEDURE [XDCL, #GATE] mmp$initiate_debug_shadowing
 0  5868      (     segment_pointer: ^cell;
 0  5869       VAR status: ost$status);
 0  5870
 0  5871      VAR
 0  5872        caller_id: ost$caller_identifier,
 0  5873        r1_pointer: ^cell,
 0  5874        r1_status: ost$status,
 0  5875        validating_ring: ost$valid_ring;
 0  5876
 0  5877      status.normal := TRUE;
 C  5878      #CALLER_ID (caller_id);
 C  5879
 C  5880 { Change the segment attributes to WRITE and flush the pages to disk.
 C  5881
 C  5882      mmp$write_modified_pages (segment_pointer, osc$maximum_offset, osc$wait, status);
44  5883      IF NOT status.normal THEN
4C  5884        RETURN;
4E  5885      IFEND;
4E  5886
4E  5887 {  Issue ring_1 call to establish ACTIVE segment.
4E  5888
4E  5889      r1_pointer := segment_pointer;
4E  5890      mmp$initiate_shadowing_r1 (r1_pointer, caller_id.ring, mmc$ssk_read_only_file, r1_status);
78  5891      IF NOT r1_status.normal THEN
80  5892        osp$set_status_condition (r1_status.condition, status);
90  5893      IFEND;
90  5894
90  5895    PROCEND mmp$initiate_debug_shadowing;
```


MMP$INITIATE_SHADOWING

```
 0  5898 {
 0  5899 {   The purpose of this request is to initiate shadowing of files.
 0  5900 {   Modified pages of the file to be shadowed are written to disk
 0  5901 {   and the segment for the file to be shadowed is transformed into
 0  5902 {   an ACTIVE segment to allow continuation of access to files during
 0  5903 {   an online dump.
 0  5904 {
 0  5905 {        MMP$INITIATE_SHADOWING (POINTER, STATUS)
 0  5906 {
 0  5907 { POINTER: (input) This parameter specifies the process virtual
 0  5908 {        address assigned to the segment.
 0  5909 {
 0  5910 { STATUS: (output) This parameter specifies the request status.
 0  5911 {
 0  5912
 0  5913   PROCEDURE [XDCL, #GATE] mmp$initiate_shadowing
 0  5914     (     segment_p: ^cell;
 0  5915      VAR status: ost$status);
 0  5916
 0  5917     VAR
 0  5918       caller_id: ost$caller_identifier,
 0  5919       r1_pointer: ^cell,
 0  5920       r1_status: ost$status;
 0  5921
 0  5922     #KEYPOINT (osk$entry, #SEGMENT (segment_p) * osk$m, mmk$initiate_shadowing);
1E  5923     status.normal := TRUE;
1E  5924     #CALLER_ID (caller_id);
1E  5925
1E  5926 { Change the segment attributes to WRITE and flush the pages to disk.
1E  5927
1E  5928     mmp$write_modified_pages (segment_p, osc$maximum_offset, osc$wait, status);
52  5929     IF NOT status.normal THEN
5A  5930       RETURN;
5C  5931     IFEND;
5C  5932
5C  5933
5C  5934 {  Issue ring_1 call to establish ACTIVE segment.
5C  5935
5C  5936     r1_pointer := segment_p;
5C  5937     mmp$initiate_shadowing_r1 (r1_pointer, #RING (segment_p), mmc$ssk_read_write_file, r1_status);
88  5938     IF NOT r1_status.normal THEN
90  5939       osp$set_status_condition (r1_status.condition, status);
A0  5940     IFEND;
A0  5941     #KEYPOINT (osk$exit, 0, mmk$initiate_shadowing);
A4  5942
A4  5943   PROCEND mmp$initiate_shadowing;
```

```
o  5946 {
o  5947 {   The purpose of this request is to open a file for segment
o  5948 { level access.
o  5949 {
o  5950 {       MMP$OPEN_FILE_SEGMENT (SFID, SEG_ATTRIBUTES_P, POINTER_KIND,
o  5951 {         VALIDATION_RING_NUMBER, CHAPTER_NUMBER, POINTER, STATUS)
o  5952 {
o  5953 { SFID: (input) This parameter specifies the system file identifier of the file
o  5954 {       to be opened as a segment.
o  5955 {
o  5956 { SEG_ATTRIBUTES_P: (input) This parameter is a pointer to an array
o  5957 {       of segment attributes to be assigned to the segment.
o  5958 {
o  5959 { POINTER_KIND: (input) This parameter specifies the type of
o  5960 {       pointer to be constructed for the segment.
o  5961 {
o  5962 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring of
o  5963 {       execution for whom the segment is being opened.  The ring
o  5964 {       number used for validation is the max of caller ring number
o  5965 {       and 'validation_ring_number'.
o  5966 {
o  5967 { POINTER: (output) This parameter is where the segment pointer
o  5968 {       is returned.  The byte offset in the PVA is set to zero.
o  5969 {
o  5970 { STATUS: (output) This parameter specifies the request status.
o  5971 {       The possible error codes are:
o  5972 {             dme$unable_to_locate_fde
o  5973 {             dme$unable_to_get_fd_lock
o  5974 {             mme$address_not_0_mod_16384
o  5975 {             mme$asid_specified
o  5976 {             mme$binding_attribute_invalid
o  5977 {             mme$contig_mem_seg_violation
o  5978 {             mme$execute_global_invalid
o  5979 {             mme$invalid_asid_specified
o  5980 {             mme$invalid_length_requested
o  5981 {             mme$invalid_pva
o  5982 {             mme$invalid_ring_brackets
o  5983 {             mme$invalid_shadow_segment
o  5984 {             mme$invalid_shared_taskid
o  5985 {             mme$length_not_0_mod_16384
o  5986 {             mme$pages_already_assigned
o  5987 {             mme$ref_to_unrecovered_file
o  5988 {             mme$ring_violation
o  5989 {             mme$segment_number_is_in_use
o  5990 {             mme$segment_number_not_in_use
o  5991 {             mme$segment_number_too_big
o  5992 {             mme$segment_origin_invalid
o  5993 {             mme$segment_table_is_full
o  5994 {             mme$software_attribute_invalid
o  5995 {             mme$unable_to_assign_contig_mem
o  5996 {             mme$unsupported_keyword
o  5997
o  5998   PROCEDURE [XDCL] mmp$open_file_segment
o  5999     (     sfid: gft$system_file_identifier,
o  6000           attr_p: ^array [ * ] of mmt$attribute_descriptor;
o  6001           pointer_kind: mmt$segment_pointer_kind;
```



```
o  6002           validation_ring_number: ost$valid_ring;
o  6003           file_limits_to_enforce: sft$file_space_limit_kind;
o  6004       VAR pointer: mmt$segment_pointer;
o  6005       VAR status: ost$status);
o  6006
o  6007     VAR
o  6008       caller_id: ost$caller_identifier,
o  6009       r1_pointer: mmt$segment_pointer,
o  6010       r1_status: ost$status,
o  6011       segment_attributes: mmt$segment_attrib_descriptor,
o  6012       validating_ring_number: ost$valid_ring;
o  6013
o  6014     #CALLER_ID (caller_id);
 C 6015     #KEYPOINT (osk$entry, 0, mmk$open_file_segment);
18 6016     status.normal := TRUE;
18 6017     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
3A 6018     segment_attributes.validating_ring_number := validating_ring_number;
3A 6019     segment_attributes.file_limits_to_enforce := file_limits_to_enforce;
3A 6020     segment_attributes.pointer_kind := pointer_kind;
3A 6021     segment_attributes.user_attributes := attr_p;
6O 6022     IF segment_attributes.user_attributes <> NIL THEN
6E 6023       PUSH segment_attributes.user_attributes: [LOWERBOUND (attr_p^) .. UPPERBOUND (attr_p^)];
A4 6024       segment_attributes.user_attributes^ := attr_p^;
E4 6025     IFEND;
E4 6026     segment_attributes.sfid := sfid;
E4 6027     mmp$build_segment (segment_attributes, NIL (shared_taskid_array) , r1_pointer, r1_status);
11C 6028    IF NOT r1_status.normal THEN
124 6029      osp$set_status_condition (r1_status.condition, status);
13A 6030    ELSE
13A 6031      pointer := r1_pointer;
148 6032    IFEND;
148 6033    #KEYPOINT (osk$exit, osk$m * #SEGMENT (pointer.cell_pointer), mmk$open_file_segment);
15E 6034
15E 6035  PROCEND mmp$open_file_segment;
```

mmp$open_segment

```
o  6037
o  6038   PROCEDURE [XDCL, #GATE] mmp$open_segment
o  6039     (     file_name: ost$name;
o  6040           seg_attributes: ^array [ * ] of mmt$attribute_descriptor;
o  6041           pointer_kind: mmt$segment_pointer_kind;
o  6042      VAR pointer: mmt$segment_pointer;
o  6043      VAR status: ost$status);
o  6044
o  6045
o  6046     VAR
o  6047       caller_id: ost$caller_identifier;
o  6048
o  6049     #CALLER_ID (caller_id);
C  6050     fmp$in_open_chapter (file_name, 0, caller_id.ring, seg_attributes, pointer_kind, pointer, status);
56 6051
56 6052   PROCEND mmp$open_segment;
```


MMP$PRESET_PAGE_STREAMING

```
o  6054 {-------------------------------------------------------------------------------------------------------------
o  6055 { PURPOSE:
o  6056 { Procedure mmp$preset_page_streaming provides the capability of presetting the SDTX of a segment so that
o  6057 { it is already in page streaming mode, with free behind TRUE and the transfer size as specified.  It returns
o  6058 { the original values of free behind and transfer size so that the caller can call again later and restore
o  6059 { the original values.
o  6060 {
o  6061 { DESIGN:
o  6062 { There is nothing fancy,  The boolean "preset_and_save_ts_fb" indicates the purpose of a call,  a value
o  6063 { of TRUE indicates preset and save the original values, a value of FALSE indicates a restore.  Nothing is
o  6064 { done to ensure calls are in order or completed.  In preset mode, the transfer size is changed if the
o  6065 { specified transfer size is > sdtx.stream.transfer_size.   IF sdtx.stream.streaming = TRUE the segment is
o  6066 { already in page streaming mode and nothing else need be done.  If sdtx.stream.streaming = FALSE then
o  6067 { the boolean sdtx.stream.preset_streaming is set to indicate that the next fault should stream.  To ensure
o  6068 { the next fault enters the page streaming code, the value of sdtx.stream.sequential_accesses is forced
o  6069 { to be  >= mmv$page_streaming_prestream.
o  6070 { When restoring the original values, the streaming boolean in the SDTX is left TRUE.  The page
o  6071 { fault process will terminate streaming if that is appropriate.  If the restore call is not made, the only
o  6072 { result is that free behind is TRUE and Transfer size =>64K which may or may not have been original values.
o  6073 {-------------------------------------------------------------------------------------------------------------
o  6074
o  6075   PROCEDURE [XDCL, #GATE] mmp$preset_page_streaming
o  6076     (     preset_and_save_ts_fb: boolean;
o  6077           pva: ^cell;
o  6078           temp_transfer_size: integer;
o  6079      VAR saved_transfer_size: 0 .. 15;
o  6080      VAR saved_free_behind: boolean;
o  6081      VAR status: ost$status);
o  6082
o  6083     VAR
o  6084       caller_id: ost$caller_identifier,
o  6085       r1_free_behind: boolean,
o  6086       r1_transfer_size: 0 .. 15,
o  6087       r1_status: ost$status;
o  6088
o  6089 {  Validate the pva and get a pointer to the segment descriptor.
o  6090
o  6091     status.normal := TRUE;
C  6092     #CALLER_ID (caller_id);
C  6093     mmp$preset_page_streaming_r1 (#SEGMENT (pva), caller_id.ring, preset_and_save_ts_fb, temp_transfer_size,
68 6094           r1_transfer_size, r1_free_behind, r1_status);
68 6095     IF NOT r1_status.normal THEN
70 6096       osp$set_status_condition (r1_status.condition, status);
84 6097     ELSE
84 6098       saved_transfer_size := r1_transfer_size;
84 6099       saved_free_behind := r1_free_behind;
9C 6100     IFEND;
9C 6101
9C 6102   PROCEND mmp$preset_page_streaming;
```

MMP$RESERVE_SEGMENT_NUMBER

```
        0  6104
        0  6105    PROCEDURE [XDCL, #GATE] mmp$reserve_segment_number
        0  6106      (    shared_stack_flag: boolean;
        0  6107       VAR segment_num_array: ^array [ * ] of ost$segment;
        0  6108       VAR status: ost$status);
        0  6109
        0  6110      VAR
        0  6111        i: integer,
        0  6112        r1_num_array_p: ^array [ * ] of ost$segment,
        0  6113        r1_status: ost$status;
        0  6114
        0  6115  { This procedure is the user interface to reserve segments for subsequent explicit assignment.
        0  6116
        0  6117      status.normal := TRUE;
        4  6118      PUSH r1_num_array_p: [LOWERBOUND (segment_num_array^) .. UPPERBOUND (segment_num_array^)];
       42  6119      FOR i := LOWERBOUND (segment_num_array^) TO UPPERBOUND (segment_num_array^) DO
       62  6120        r1_num_array_p^ [i] := 0;
       62  6121      FOREND;
       7C  6122
       7C  6123      mmp$reserve_segment_number_r1 (shared_stack_flag, r1_num_array_p, r1_status);
       A0  6124      IF NOT r1_status.normal THEN
       A8  6125        osp$set_status_condition (r1_status.condition, status);
       BC  6126      ELSE
       BC  6127        FOR i := LOWERBOUND (segment_num_array^) TO UPPERBOUND (segment_num_array^) DO
       D6  6128          segment_num_array^ [i] := r1_num_array_p^ [i];
       D6  6129        FOREND;
      106  6130      IFEND;
      106  6131
      106  6132    PROCEND mmp$reserve_segment_number;
```


MMP$REVERIFY_ACCESS

```
        0  6135
        0  6136  {
        0  6137  {   This request can be used to determine whether a PVA
        0  6138  { can be accessed by a program without causing an access
        0  6139  { violation or segment fault.
        0  6140  {
        0  6141  {   This request is similar to MMP$VERIFY_ACCESS but runs
        0  6142  { faster. The speed improvement is at a cost of less checking.
        0  6143  { The request simply verifies that the requested segment is still
        0  6144  { valid in the segment table. Checking of access rights and
        0  6145  { read-beyond-eoi is NOT done.
        0  6146  {
        0  6147  {       MMP$REVERIFY_ACCESS (PVA_P): BOOLEAN
        0  6148  {
        0  6149  { PVA_P: (input) This parameter is a pointer to the PVA to be
        0  6150  {       tested.
        0  6151  {
        0  6152  { BOOLEAN: (output) The boolean result of this function specifies
        0  6153  {       whether access is valid.
        0  6154  {
        0  6155
        0  6156    FUNCTION [XDCL] mmp$reverify_access
        0  6157      (    pva_p: ^^cell): boolean;
        0  6158
        0  6159      VAR
        0  6160        sdt_entry_p: ^mmt$segment_descriptor,
        0  6161        sdtx_entry_p: ^mmt$segment_descriptor_extended,
        0  6162        segnum: ost$segment,
        0  6163        xcb_p: ^ost$execution_control_block;
        0  6164
        0  6165      segnum := #SEGMENT (pva_p^);
        4  6166
        4  6167      xcb_p := #ADDRESS (1, osc$segnum_job_fixed_heap, #READ_REGISTER (osc$pr_base_constant));
       18  6168      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segnum);
       18  6169      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
       18  6170      mmp$reverify_access := (segnum <= xcb_p^.xp.segment_table_length) AND
       A6  6171           (sdt_entry_p^.ste.v1 <> osc$v1_invalid_entry) AND (sdtx_entry_p^.access_state <>
       A6  6172           mmc$sas_terminate_access);
       A6  6173    FUNCEND mmp$reverify_access;
        0  6174
```

MMP$SET_ACCESS_SELECTIONS

```
 0  6176 {
 0  6177 {  The purpose of this request is to change the segment access selections
 0  6178 {  associated with a segment. Three access selections are currently supported
 0  6179 {  for access to a segment: sequential, random, and read_tu. Sequential access
 0  6180 {  should be selected if access to the segment is sequential and more than 4
 0  6181 {  pages will be accessed sequentially. Sequential access should NOT be selected
 0  6182 {  for segments that are managed via ADVISE requests. Read_tu access should be
 0  6183 {  selected if the task is randomly accessing large blocks of data in the
 0  6184 {  segment, or if the task is sequentially accessing multiple parts of the
 0  6185 {  segment.  Random access should be selected otherwise.
 0  6186 {
 0  6187 {  An access selection of sequential causes memory manager to read an entire
 0  6188 {  transfer unit when a page fault occurs for a page on disk. In addition
 0  6189 {  pages assigned to the segment may be automatically removed when new
 0  6190 {  pages are added as a result of a page fault.
 0  6191 {
 0  6192 {  An access selection of read_tu causes memory manager to read multiple pages,
 0  6193 {  one or more transfer units, when a page fault occurs for a page on disk.  In
 0  6194 {  addition pages assigned to the segment are not automatically removed when new
 0  6195 {  pages are added as a result of a page fault.
 0  6196 {
 0  6197 {  An access selection of random causes memory manager to read one page for each
 0  6198 {  page fault for a page on disk.  No pages pages are removed as a result of the
 0  6199 {  page fault.
 0  6200 {
 0  6201 {           MMP$SET_ACCESS_SELECTIONS (PVA, ACCESS_SELECTIONS, STATUS)
 0  6202 {
 0  6203 {    PVA: (input) This parameter specifies the segment.
 0  6204 {
 0  6205 {    ACCESS_SELECTIONS: (input) This parameter specifies the access
 0  6206 {         selections for the segment.
 0  6207 {
 0  6208 {    STATUS: (output) This parameter specifies the request status.
 0  6209 {         The possible error codes are:
 0  6210 {              dme$unable_to_locate_fde
 0  6211 {              dme$unable_to_get_fd_lock
 0  6212 {              mme$invalid_ring_brackets
 0  6213 {              mme$ring_violation
 0  6214 {              mme$segment_number_not_in_use
 0  6215 {              mme$segment_number_too_big
 0  6216 {              mme$segment_origin_change
 0  6217 {
 0  6218
 0  6219   PROCEDURE [XDCL, #GATE] mmp$set_access_selections
 0  6220     (     pva: ^cell;
 0  6221           access_selections: mmt$access_selections;
 0  6222      VAR status: ost$status);
 0  6223
 0  6224     VAR
 0  6225       caller_id: ost$caller_identifier,
 0  6226       r1_status: ost$status,
 0  6227       validating_ring_number: ost$valid_ring;
 0  6228
 0  6229     #KEYPOINT (osk$entry, osk$m * #SEGMENT (pva), mmk$set_access_selections);
1E  6230
1E  6231     status.normal := TRUE;
```


MMP$SET_ACCESS_SELECTIONS

```
1E  6232     #CALLER_ID (caller_id);
1E  6233     determine_validating_ring_num (caller_id.ring, #RING (pva), validating_ring_number);
48  6234     mmp$set_access_selections_r1 (#SEGMENT (pva), validating_ring_number, access_selections, r1_status);
7A  6235     IF NOT r1_status.normal THEN
82  6236       osp$set_status_condition (r1_status.condition, status);
94  6237     IFEND;
94  6238
94  6239     #KEYPOINT (osk$exit, 0, mmk$set_access_selections);
98  6240
98  6241   PROCEND mmp$set_access_selections;
```

MMP$SET_SEGMENT_LENGTH

```
o  6244 {
o  6245 {   The purpose of this procedure is to explicitly set the segment length.
o  6246 { Segment length is defined as the last byte in the segment that can be
o  6247 { referenced, references beyond segment length are treated the same as a
o  6248 { read beyond EOI.
o  6249 {
o  6250 {         MMP$SET_SEGMENT_LENGTH (PVA, VALIDATION_RING_NUMBER, SEGMENT_LENGTH,
o  6251 {           STATUS)
o  6252 {
o  6253 { PVA: (input) This parameter specifies the segment for which segment length
o  6254 {         is being set.
o  6255 {
o  6256 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring number
o  6257 {         used for validation if it is greater than the caller ring number.
o  6258 {
o  6259 { SEGMENT_LENGTH: (input) This parameter specifies the new segment length.
o  6260 {
o  6261 { STATUS: (output) This parameter is where the request status is returned
o  6262 {         to the caller.  The possible error codes are:
o  6263 {               mme$caller_not_in_write_bracket
o  6264 {               mme$invalid_pva
o  6265 {               mme$no_write_access
o  6266 {               mme$ref_to_unrecovered_file
o  6267 {               mme$segment_number_not_in_use
o  6268 {               mme$segment_number_too_big
o  6269 {
o  6270
o  6271   PROCEDURE [XDCL, #GATE] mmp$set_segment_length
o  6272     (     pva: ^cell;
o  6273           validation_ring_number: ost$valid_ring;
o  6274           segment_length: ost$segment_length;
o  6275      VAR status: ost$status);
o  6276
o  6277     VAR
o  6278       caller_id: ost$caller_identifier,
o  6279       r1_status: ost$status,
o  6280       segment_pointer: ^cell,
o  6281       validating_ring_number: ost$valid_ring;
o  6282
o  6283     #KEYPOINT (osk$entry, 0, mmk$set_segment_length);
10 6284     #CALLER_ID (caller_id);
10 6285     status.normal := TRUE;
10 6286     determine_validating_ring_num (caller_id.ring, validation_ring_number, validating_ring_number);
3A 6287     segment_pointer := pva;
3A 6288     mmp$set_segment_length_r1 (#SEGMENT (segment_pointer), validating_ring_number, segment_length, r1_status);
6C 6289     IF NOT r1_status.normal THEN
74 6290       osp$set_status_condition (r1_status.condition, status);
86 6291     IFEND;
86 6292     #KEYPOINT (osk$exit, osk$m * #SEGMENT (pva), mmk$set_segment_length);
98 6293
98 6294   PROCEND mmp$set_segment_length;
```


MMP$STORE_SEGMENT_ATTRIBUTES

```
o  6297 {
o  6298 {   The purpose of this request is to allow a user to change
o  6299 { attributes of a segment.  Some attribute changes (such as
o  6300 { changing a segment attribute to BINDING) are rejected unless
o  6301 { the calling procedure is running within a certain ring bracket.
o  6302 {
o  6303 {       MMP$STORE_SEGMENT_ATTRIBUTES (PVA, VALIDATION_RING_NUMBER,
o  6304 {         SEG_ATTRIBUTES, STATUS)
o  6305 {
o  6306 { PVA: (input) This parameter specifies the segment to be changed.
o  6307 {
o  6308 { VALIDATION_RING_NUMBER: (input) This parameter specifies the ring
o  6309 {       of execution for whom the change is being made.  The
o  6310 {       ring number used for validation is the max of caller ring
o  6311 {       number and 'validation_ring_number'.
o  6312 {
o  6313 { SEG_ATTRIBUTES: (input) This parameter is an adaptable array of
o  6314 {       segment attribute descriptors.  Each attribute descriptor specifies
o  6315 {       a new value for a segment attribute.
o  6316 {
o  6317 { STATUS: (output) This parameter specifies the request status.
o  6318 {       The possible error codes are:
o  6319 {             dme$unable_to_get_fd_lock
o  6320 {             dme$unable_to_locate_fde
o  6321 {             mme$address_not_0_mod_16384
o  6322 {             mme$asid_specified
o  6323 {             mme$binding_attribute_invalid
o  6324 {             mme$execute_global_invalid
o  6325 {             mme$execute_local_invalid
o  6326 {             mme$invalid_asid_specified
o  6327 {             mme$invalid_ring_brackets
o  6328 {             mme$invalid_shadow_segment
o  6329 {             mme$length_not_0_mod_16384
o  6330 {             mme$ring_violation
o  6331 {             mme$segment_number_is_in_use
o  6332 {             mme$segment_number_not_in_use
o  6333 {             mme$segment_number_too_big
o  6334 {             mme$segment_origin_change
o  6335 {             mme$segment_orgin_invalid
o  6336 {             mme$set_unmodifiable_attribute
o  6337 {             mme$software_attribute_invalid
o  6338 {             mme$unsupported_keyword
o  6339 {
o  6340
o  6341
o  6342   PROCEDURE [XDCL, #GATE] mmp$store_segment_attributes
o  6343     (     pva: ^cell;
o  6344           validation_ring_number: ost$valid_ring;
o  6345           attr: array [ * ] of mmt$attribute_descriptor;
o  6346      VAR status: ost$status);
o  6347
o  6348     VAR
o  6349       caller_id: ost$caller_identifier,
o  6350       r1_status: ost$status,
o  6351       r1_segment_attributes_p: ^array [ * ] of mmt$attribute_descriptor;
o  6352
```

MMP$STORE_SEGMENT_ATTRIBUTES

```
 0  6353      #KEYPOINT (osk$exit, 0, mmk$store_segment_attributes);
10  6354
10  6355      status.normal := TRUE;
10  6356      #CALLER_ID (caller_id);
10  6357      PUSH r1_segment_attributes_p: [LOWERBOUND (attr) .. UPPERBOUND (attr)];
54  6358      r1_segment_attributes_p^ := attr;
9A  6359      mmp$store_segment_attributes_r1 (#SEGMENT (pva), caller_id.ring, r1_segment_attributes_p, r1_status);
CC  6360      IF NOT r1_status.normal THEN
D4  6361        osp$set_status_condition (r1_status.condition, status);
E6  6362      IFEND;
E6  6363
E6  6364      #KEYPOINT (osk$exit, 0, mmk$store_segment_attributes);
EA  6365
EA  6366    PROCEND mmp$store_segment_attributes;
```


MMP$TERMINATE_SHADOWING

```
 0  6369 {
 0  6370 {    The purpose of this request is to terminate shadowing of files.
 0  6371 {    o If required (UPDATE parameter set to true), a request will be issued to
 0  6372 {    update
 0  6373 {      the PASSIVE file with the ACTIVE .
 0  6374 {    o A request to mmp$terminate_shadowing_r1 will be issued to destroy the
 0  6375 {    ACTIVE
 0  6376 {      file and to activate the PASSIVE file again.
 0  6377 {
 0  6378 {        MMP$TERMINATE_SHADOWING (POINTER, UPDATE, STATUS)
 0  6379 {
 0  6380 { POINTER: (input) This parameter specifies the process virtual
 0  6381 {        address assigned to the ACTIVE segment.
 0  6382 {
 0  6383 { UPDATE: (input) This boolean parameter specifies if an update
 0  6384 {        (PASSIVE with ACTIVE)  is required or not.
 0  6385 {
 0  6386 { STATUS: (output) This parameter specifies the request status.
 0  6387 {
 0  6388
 0  6389    PROCEDURE [XDCL, #GATE] mmp$terminate_shadowing
 0  6390      (     segment_p: ^cell;
 0  6391            update: boolean;
 0  6392       VAR status: ost$status);
 0  6393
 0  6394      VAR
 0  6395        access_selections: mmt$access_selections,
 0  6396        dest_p: mmt$segment_pointer,
 0  6397        r1_status: ost$status;
 0  6398
 0  6399      #KEYPOINT (osk$entry, #SEGMENT (segment_p) * osk$m, mmk$terminate_shadowing);
16  6400      status.normal := TRUE;
16  6401
16  6402 {  Determine if update (PASSIVE with ACTIVE)is required.
16  6403
16  6404      IF update THEN
28  6405        update_passive_with_active (segment_p, status);
38  6406      IFEND;
38  6407
38  6408      mmp$terminate_shadowing_r1 (#SEGMENT (segment_p), r1_status);
5A  6409      IF NOT r1_status.normal THEN
62  6410        osp$set_status_condition (r1_status.condition, status);
74  6411      IFEND;
74  6412      #KEYPOINT (osk$exit, 0, mmk$terminate_shadowing);
78  6413
78  6414    PROCEND mmp$terminate_shadowing;
```

```
        o  6417 {
        o  6418 {   This request can be used to determine whether a PVA
        o  6419 { can be accessed by a program without causing an access
        o  6420 { violation or segment fault.
        o  6421 {
        o  6422 {       MMP$VERIFY_ACCESS (PVA_P, ACCESS_MODE): BOOLEAN
        o  6423 {
        o  6424 { PVA_P: (input) This parameter is a pointer to the PVA to be
        o  6425 {       tested.
        o  6426 {
        o  6427 { ACCESS_MODE: (input) This parameter specifies the type of
        o  6428 {       access to be tested.
        o  6429 {
        o  6430 { BOOLEAN: (output) The boolean result of this function specifies
        o  6431 {       whether access is valid.
        o  6432 {
        o  6433
        o  6434   FUNCTION [XDCL, #GATE] mmp$verify_access
        o  6435     (     pva_p: ^^cell;
        o  6436           access_mode: mmt$va_access_mode): boolean;
        o  6437
        o  6438     TYPE
        o  6439       external_code_base_pointer = packed record
        o  6440         fill: 0 .. 0ff(16),
        o  6441         vmid: 0 .. 0ff(16),
        o  6442         xp: boolean,
        o  6443         fill2: 0 .. 7,
        o  6444         r3: 0 .. 15,
        o  6445         code_pva: ost$pva,
        o  6446         fill3: 0 .. 0ffff(16),
        o  6447         binding_pva: ost$pva,
        o  6448       recend;
        o  6449
        o  6450     VAR
        o  6451       caller_id: ost$caller_identifier,
        o  6452       code_pva: ost$pva,
        o  6453       pointer: record
        o  6454         case (pva, code_pointer, str) of
        o  6455         = pva =
        o  6456           pva: ost$pva,
        o  6457         = code_pointer =
        o  6458           cbp_p: ^external_code_base_pointer,
        o  6459           static_link: ost$pva,
        o  6460         = str =
        o  6461           s: string (12),
        o  6462         casend,
        o  6463       recend,
        o  6464       ptr: record
        o  6465         case 0 .. 1 of
        o  6466         = 0 =
        o  6467           pva_p: ^^cell,
        o  6468         = 1 =
        o  6469           s_p: ^string (12),
        o  6470         casend,
        o  6471       recend,
        o  6472       ref_r: ost$ring,
```



```
        o  6473       sd_p: ^mmt$segment_descriptor,
        o  6474       sdtx_p: ^mmt$segment_descriptor_extended,
        o  6475       segnum: ost$segment,
        o  6476       status: ost$status,
        o  6477       xcb_p: ^ost$execution_control_block;
        o  6478
        o  6479     mmp$verify_access := TRUE;
        C  6480     ptr.pva_p := pva_p;
        C  6481     pointer.s := ptr.s_p^;
       26  6482     segnum := pointer.pva.seg;
       26  6483     #KEYPOINT (osk$entry, segnum * osk$m, mmk$verify_access);
       36  6484
       36  6485   /verify_access/
       36  6486     BEGIN
       36  6487       xcb_p := #ADDRESS (1, osc$segnum_job_fixed_heap, #READ_REGISTER (osc$pr_base_constant));
       3C  6488       IF segnum > xcb_p^.xp.segment_table_length THEN
       58  6489         mmp$verify_access := FALSE;
       58  6490         EXIT /verify_access/;
       5E  6491       IFEND;
       5E  6492       sd_p := mmp$get_sdt_entry_p (xcb_p, segnum);
       5E  6493       sdtx_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
       5E  6494       IF sd_p^.ste.v1 = osc$v1_invalid_entry THEN
       AE  6495         mmp$verify_access := FALSE;
       AE  6496         EXIT /verify_access/;
       B4  6497       IFEND;
       B4  6498       IF (sdtx_p^.access_state = mmc$sas_terminate_access) THEN
       C0  6499         mmp$verify_access := FALSE;
       C0  6500         EXIT /verify_access/;
       C8  6501       IFEND;
       C8  6502
       C8  6503
       C8  6504       #CALLER_ID (caller_id);
       C8  6505       ref_r := pointer.pva.ring;
       C8  6506       IF ref_r = 0 THEN
       DC  6507         mmp$verify_access := FALSE;
       DC  6508         EXIT /verify_access/;
       E4  6509       IFEND;
       E4  6510
       E4  6511
       E4  6512 {  Move pointer being verified to a local variable so that ring number checking will be
       E4  6513 {  valid on recursive calls.
       E4  6514
       E4  6515
       E4  6516       IF caller_id.ring > ref_r THEN
       EC  6517         ref_r := caller_id.ring;
       EC  6518         pointer.pva.ring := caller_id.ring;
       FC  6519       IFEND;
       FC  6520
       FC  6521       CASE access_mode OF
      132  6522       = mmc$va_read =
      132  6523         mmp$verify_access := (ref_r <= sd_p^.ste.r2) AND (sd_p^.ste.rp <> osc$non_readable);
      15A  6524       = mmc$va_write =
      15A  6525         mmp$verify_access := (ref_r <= sd_p^.ste.r1) AND (sd_p^.ste.wp <> osc$non_writable);
      182  6526       = mmc$va_read_write =
      182  6527         mmp$verify_access := (ref_r <= sd_p^.ste.r1) AND (sd_p^.ste.wp <> osc$non_writable) AND
      1BC  6528               (sd_p^.ste.rp <> osc$non_readable);
```

```
1BC  6529          = mmc$va_execute =
1BC  6530            mmp$verify_access := (ref_r >= sd_p^.ste.r1) AND (ref_r <= sd_p^.ste.r2) AND
1F4  6531                  (sd_p^.ste.xp <> osc$non_executable);
1F4  6532          = mmc$va_read_execute =
1F4  6533            mmp$verify_access := mmp$verify_access (#LOC (pointer.pva), mmc$va_execute) AND
230  6534                  mmp$verify_access (#LOC (pointer.pva), mmc$va_read);
230  6535          = mmc$va_binding =
230  6536            mmp$verify_access := sd_p^.ste.rp = osc$binding_segment;
246  6537          = mmc$va_pointer_to_procedure =
246  6538
246  6539 {  To verify a pointer to procedure the following must be checked:
246  6540 {     . The procedure pointer must be in a segment with read access.
246  6541 {     . The static link pointer, code base pointer, code PVA, or binding PVA must not have a ring number
246  6542 {       equal to zero.
246  6543 {     . The code base pointer must be in a segment with the "binding" attribute and be in a ring
246  6544 {       readable segment.
246  6545 {     . The caller must be within the call bracket.
246  6546 {     . The code PVA in the code base pointer must be in a segment with "execute" privilege.
246  6547 {     . The binding PVA in the code base pointer must be in a segment with the "binding" attribute
246  6548 {       if this is a two word external code base pointer.
246  6549
246  6550           IF mmp$verify_access (#LOC (pointer.pva), mmc$va_read) = FALSE THEN
266  6551             mmp$verify_access := FALSE;
266  6552             EXIT /verify_access/;
26E  6553           IFEND;
26E  6554
26E  6555           IF pointer.static_link.ring = 0 THEN
27A  6556             mmp$verify_access := FALSE;
27A  6557             EXIT /verify_access/;
282  6558           IFEND;
282  6559
282  6560           IF pointer.cbp_p^.code_pva.ring = 0 THEN
292  6561             mmp$verify_access := FALSE;
292  6562             EXIT /verify_access/;
29A  6563           IFEND;
29A  6564
29A  6565           pointer.pva.ring := ref_r;
29A  6566           IF (mmp$verify_access (#LOC (pointer.pva), mmc$va_binding) AND mmp$verify_access (#LOC (pointer.pva),
2D8  6567                 mmc$va_read)) = FALSE THEN
2D8  6568             mmp$verify_access := FALSE;
2D8  6569             EXIT /verify_access/;
2E0  6570           IFEND;
2E0  6571
2E0  6572           IF ref_r > pointer.cbp_p^.r3 THEN
2F0  6573             mmp$verify_access := FALSE;
2F0  6574             EXIT /verify_access/;
2F8  6575           IFEND;
2F8  6576
2F8  6577 {  The caller is within the call bracket so the call is possible.  The ring of execution will be r2 from
2F8  6578 {  the code pva segment descriptor if it is greater than the caller's ring number, if not the caller's
2F8  6579 {  ring number is the ring of execution.  The ring of execution is used as the ring number in
2F8  6580 {  validating the code pva and the new binding pva if there is one.
2F8  6581
2F8  6582           code_pva := pointer.cbp_p^.code_pva;
2F8  6583           IF ref_r > sd_p^.ste.r2 THEN
310  6584             code_pva.ring := sd_p^.ste.r2;
```



```
310  6585             ref_r := sd_p^.ste.r2;
328  6586           ELSE
328  6587             code_pva.ring := ref_r;
334  6588           IFEND;
334  6589
334  6590           IF mmp$verify_access (#LOC (code_pva), mmc$va_execute) = FALSE THEN
352  6591             mmp$verify_access := FALSE;
352  6592             EXIT /verify_access/;
35A  6593           IFEND;
35A  6594
35A  6595           IF pointer.cbp_p^.xp = TRUE THEN
368  6596             IF pointer.cbp_p^.binding_pva.ring = 0 THEN
374  6597               mmp$verify_access := FALSE;
374  6598               EXIT /verify_access/;
37C  6599             IFEND;
37C  6600           IFEND;
380  6601         ELSE
380  6602           mmp$verify_access := FALSE;
384  6603         CASEND;
384  6604       END /verify_access/;
384  6605
384  6606       #KEYPOINT (osk$exit, 0, mmk$verify_access);
388  6607
388  6608     FUNCEND mmp$verify_access;
```

MMP$VOLUME_UNAVAILABLE_FLAG_HDL

```
        0  6611
        0  6612   PROCEDURE [XDCL] mmp$volume_unavailable_flag_hdl
        4  6613     (     flag_id: ost$system_flag);
        4  6614
        4  6615     bap$exit_fap_on_condition (mme$volume_unavailable);
       1C  6616     osp$wait_for_unavailable_volume (mme$volume_unavailable);
       30  6617
       30  6618   PROCEND mmp$volume_unavailable_flag_hdl;
```


MMP$WAIT_FOR_UNAVAILABLE_VOLUME

```
        0  6620
        0  6621   PROCEDURE [XDCL, #GATE] osp$wait_for_unavailable_volume
        C  6622     (     condition: ost$status_condition_code);
        C  6623
        C  6624     osp$verify_system_privilege;
       42  6625
       42  6626     osp$wait_on_condition (condition);
       56  6627
       56  6628   PROCEND osp$wait_for_unavailable_volume;
        0  6630 MODEND mmm$segment_manager_job_temp;

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

   **** NO DIAGNOSTICS
```

IDENTIFIER	DEFINED ON LINE	REFERENCES						
access_control	2185	5369/M						
access_control	2376	5369						
access_mode	6436	6521						
access_selections	4586	4617/M	4619/M	4621/M	4718/P			
access_selections	4996	5025						
access_selections	5315	5412						
access_selections	6221	6234/P						
access_state	2931	6171	6498					
addr	391	4685	4707	4708	4714			
addr_list	5771	5786	5786	5792/M				
addr_returned	4588	4672/P	4678	4679	4714/S	4714/S		
addr_returned	5772	5794/M						
address_list	4587	4672/P	4684	4685	4707	4708	4709/P	4710/P	4711/P
		4714	4714					
amc$cell_pointer	9	13	5010	5029	5395	5416		
amc$file_byte_limit	1185	1188	1190					
amc$heap_pointer	9	15						
amc$sequence_pointer	10	17	5012	5031	5397	5421		
amt$file_byte_address	1188	1149	4133	5623	5624			
amt$file_limit	1190	1153						
amt$local_file_name	3788	3777						
amt$pointer_kind	9	12	4995	5314				
amt$segment_pointer	11	4795	4798	4803	4997	5316	5480	5582
attr	6345	6357	6357	6358				
attr_p	6000	6021	6023	6023	6024			
bad_offset	4259	4277						
bad_offset	6621	6624						
bap$exit_fap_on_condition	3742	5660	6615					
binding_pva	6447	6596						
byte	5320	5451/M						
caller_id	4748	4753	4754/P					
caller_id	4802	4808	4809/P					
caller_id	4843	4847	4849/P					
caller_id	4888	4892	4895/P					
caller_id	4930	4933	4935/P					
caller_id	5001	5007	5017					
caller_id	5104	5110	5113/P					
caller_id	5170	5185	5186					
caller_id	5235	5240	5242					
caller_id	5321	5402	5404					
caller_id	5484	5488	5494/P					
caller_id	5541	5545	5548/P					
caller_id	5586	5589	5591/P					
caller_id	5746	5754	5757/P					
caller_id	5777	5785	5787/P					
caller_id	5833	5841	5842/P					
caller_id	5872	5878	5890/P					
caller_id	5918	5924						
caller_id	6008	6014	6017/P					
caller_id	6047	6049	6050/P					
caller_id	6084	6092	6093/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
caller_id	6225	6232	6233/P						
caller_id	6278	6284	6286/P						
caller_id	6349	6356	6359/P						
caller_id	6451	6504	6516	6517	6518				
caller_ring	4564	4568	4569						
caller_ring	4742	4754	4754						
caller_ring	4794	4809	4809						
caller_ring	4882	4895	4895						
caller_ring	5095	5113	5113						
caller_ring	5535	5548	5548						
caller_ring	5826	5842	5842						
caller_ring	5998	6017	6017						
caller_ring	6219	6233	6233						
caller_ring	6271	6286	6286						
cbp_p	6458	6560	6572	6582	6595	6596			
cell_pointer	14	4810/P	4814/M	4814	4815	5030/M	5420/M	5434/P	5489
		5492	5494/P	5497/M	5591/P	5594/M			
cell_pointer	2230	4654	4708	4893	4896/P	4901/M	4935/P	4939/M	5026/P
		5030	5413/P	5417	5419	5420	5492/M	5546	5549/P
		5552/M	6033						
code_pointer	6454	6457							
code_pva	6445	6560	6582						
code_pva	6452	6582/M	6584/M	6587/M	6590/P				
condition	2292	4227/M	4757/P	4757/M	4817/P	4817/M	4851/P	4851/M	4899/P
		4899/M	4937/P	4937/M	5037/P	5037/M	5126/P	5126/M	5196/P
		5196/M	5254/P	5254/M	5456/P	5456/M	5499/P	5499/M	5554/P
		5554/M	5596/P	5596/M	5642	5655	5655	5656	5709/P
		5709/M	5759/P	5759/M	5790/P	5790/M	5845/P	5845/M	5892/P
		5892/M	5939/P	5939/M	6029/P	6029/M	6096/P	6096/M	6125/P
		6125/M	6236/P	6236/M	6290/P	6290/M	6361/P	6361/M	6410/P
		6410/M							
condition	4222	4227							
condition	4742	4757							
condition	4794	4817							
condition	4838	4851							
condition	4882	4899							
condition	4924	4937							
condition	4994	5037							
condition	5095	5126							
condition	5161	5196							
condition	5227	5254							
condition	5312	5456							
condition	5479	5499							
condition	5535	5554							
condition	5581	5596							
condition	5691	5709							
condition	5740	5759							
condition	5768	5790							
condition	5826	5845							
condition	5867	5892							
condition	5913	5939							
condition	5998	6029							
condition	6075	6096							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
condition	6105	6125
condition	6219	6236
condition	6271	6290
condition	6342	6361
condition	6389	6410
condition	6622	6626/P
contiguous_flag	5322	5342/M 5378/M 5433 5444
contiguous_real_memory	2379	5377
conv_ptr	5171	5182/M 5191
dest	3842	3852
dest	4578	4655 4709
dest	4590	4654/M 4655/P 4656/P 4708/M 4709/P 4711/P
dest_p	4589	4630/P 4654 4708 4719/P
determine_validating_ring_num	4563	4574 4754 4809 4895 5113 5548 5842 6017
		6233 6286
dfc$min_cdcnet_errors	320	326 328 331 334 337 340 343 346
		349 352 355 375
dfc$min_driver_test_errors	315	363 366 369 372
dfc$min_ecc	25	31 34 37 40 43 47 51 54
		57 60 63 66 69 72 75 78
		81 84 88 91 94 98 101 104
		107 110 113 116 119 122 125 128
		131 134 137 141 145 148 151 155
		158 161 164 167 170 173 176 179
		183 186 190 193 196 199 202 205
		208 212 216 220 223 226 229 233
		237 240 243 246 249 252 256 259
		263 266 269 272 275 278 282 285
		288 292 296 301 304 308 315 317
		320 322
dfc$min_mm_recovery_errors	322	378 382 385
dfe$family_not_served	190	5655
dfe$server_has_terminated	173	5656
dfe$server_not_active	122	5655
dm_element_length	4591	4684/M 4687 4687 4699/M 4699
dm_element_offset	4592	4685/M 4695 4695 4698/M 4698
dmc$device_manager_error_code	475	476 479 482 485 488 491 494 497
		500 503 506 509 512 515 518 521
		524 527 530 533 536 539 542 545
		548 551 554 557 560 563 566 569
		572 575 578 581 584 587 590 593
		596 599 602 605 608 611 614 617
		620 623 626 629 632 635 638 641
		644 647 650 653 656 659 662 665
		668 671 674 677 680 683 686 689
		692 695 698 701 704 707 710 713
		716 719 722 725 728 731 734 737
		740 743 746 749 752 755 758 761
		764 767 770 773 776 779 782 785
		788 791 794 797 800 803 806 809
		812 815 818 821 824 827 830 833
		836 839 842 845 848 851 854 857

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
		860 863 866 869 872 875 878 881
		884 887 890 893 896 899 902 905
		908 911 914 917 920 923 926 929
		932 935 938 941 944 947 950 953
		956 959 962 965 968 971 974 977
		980 983 986 989 992 995 998 1001
		1004 1007 1010 1013 1016 1019 1022 1025
		1028 1031 1034 1037 1043 1046 1049 1052
		1055 1058 1061 1064 1067 1070 1073 1076
		1079 1082 1086 1089 1092 1097 1100 1103
		1106 1109 1112 1115 1118 1121 1124
dme$unable_to_alloc_all_space	962	5642 5660/P 5668/P
dmp$get_initialized_addresses	3755	4672
dmt$addr_length_pair	390	3757 3973 4587 5771 5779
dmt$chapter_number	3796	3778
dmt$system_file_id	3769	3755
dpp$put_critical_message	3748	5664
dsp$system_committed	3773	5663
external_code_base_pointer	6439	6458
fde_p	3800	3815/M 3816 3817/M
fde_p	4578	4667/M 4667 4667/M
fde_p	4593	4667/P 4668
file	5769	5787/P
file_entry_index	1495	3815 4667
file_hash	1146	3816 4667
file_hash	1497	3816 4667
file_limits_to_enforce	2344	5018/M 5115/M 5187/M 5243/M 5405/M 6019/M
file_limits_to_enforce	6003	6019
file_name	6039	6050/P
fmp$in_open_chapter	3777	6050
get_list_of_addresses	4670	4670 4716
gfc$fde_size	3832	3815 4667
gfc$fde_table_base	3830	3815 3831 4667
gfc$fk_catalog	1225	1237
gfc$fk_job_local_file	1227	1236
gfc$fm_mass_storage_file	1240	1162 4668
gfc$fm_served_file	1241	1165
gfc$tr_job	1505	3809 4667
gfc$tr_system	1505	3808 4667
gfp$get_fde_p	3799	3820 4667
gft$allocation_unit_size	1196	1151
gft$attach_count	1201	1142 1143
gft$fde_flags	1171	1139
gft$file_desc_entry_p	3837	3800 4593
gft$file_descriptor_entry	1136	1128 1141 3837
gft$file_descriptor_index	1210	1495
gft$file_kind	1221	1145 1233
gft$file_media	1240	1161
gft$locked_file_desc_entry_p	1128	3943
gft$open_count	1270	1144 1286

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gft$queue_status	1281	1154							
gft$segment_lock_info	1285	1147							
gft$signature_lock	1246	1137							
gft$system_file_identifier	1494	2346	2535	2932	3024	3768	3798	3951	4526
		5998							
gft$table_residence	1505	1496							
gft$transfer_unit_size	1207	1152							
gfv$null_sfid	4526	5021	5118	5193	5251	5408	5706/P		
heap_pointer	16	5034/M	5430/M						
heap_pointer	2234	5034	5427	5429	5430				
i	4595	4652	4653/S	4654/S	4679	4684/S	4685/S	4707/S	4708/S
		4709/S	4710/S	4711/S					
i	5325	5350	5351/S	5353/S	5354/S	5354/S	5355/S	5355/S	5357/S
		5358/S	5358/S	5360/S	5361/S	5361/S	5363	5365/S	5366/S
		5366/S	5368/S	5369/S	5369/S	5371/S	5372/S	5376/S	5376/S
		5377/S	5381	5383/S					
i	5622	5678	5678/S						
i	6111	6119	6120/S	6127	6128/S	6128/S			
i#move	3841	3857	4655	4709					
id	4262	4269	4273/S						
id	6621	6624	6624/S						
increment	5326	5449/M	5450	5450	5451/S	5452/M	5452		
iot$transfer_count	3098	3086							
jmc$highest_prio_age_interval	2780	2771	2781						
jmc$highest_service_accumulator	2812	2813							
jmc$highest_service_factor_valu	2864	2857							
jmc$keyword_offset_maximum	2797	2772							
jmc$kill_job_flag	3665	5671							
jmc$kjl_maximum_entries	2745	2738	2739						
jmc$kol_maximum_entries	2755	2740							
jmc$logout_flag_id	3665	5671							
jmc$max_active_jobs	2736	2723	2731	2732					
jmc$max_ajl_ord	2737	2736							
jmc$max_dispatching_control	2681	2685							
jmc$max_dispatching_priority	2582	2542	2545	2546					
jmc$maximum_job_count	2752	2745							
jmc$maximum_output_count	2762	2755							
jmc$maximum_service_classes	2830	2833							
jmc$min_dispatching_control	2680	2684							
jmc$null_service_class	2823	2824							
jmc$priority_aging_interval_max	2771	2768							
jmc$priority_p1	2596	2543							
jmc$priority_p10	2605	2544							
jmc$priority_p14	2609	2544							
jmc$priority_p8	2603	2543							
jmc$reserved_ajls	2741	2736							
jmc$service_accumulator_maximum	2804	2801							
jmc$service_factor_value_max	2857	2854							
jmc$system_default_offset	2796	2797							
jmc$terminate_job_flag	3664	5670							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$unlimited_offset	2793	2782	2814						
jmt$dispatching_control	2651	2634							
jmt$dispatching_control_index	2684	2651							
jmt$dispatching_controls	2654	2652							
jmt$dispatching_priority	2542	2507	2509	2656					
jmt$job_priority	2712	2643	2644	2645	2646				
jmt$maximum_active_jobs	2723	2628							
jmt$priority_aging_interval	2768	2636							
jmt$scheduling_priority	2642	2635							
jmt$service_accumulator	2801	2626	2627						
jmt$service_class_index	2833	2619	2629						
jmt$service_class_name	2836	2621	2622						
jmt$service_factor_value	2854	2630							
jmt$service_factors	2850	2630							
jmt$task_time_slice	2694	2674	2675						
jmt$time_slice_values	2673	2520	2658						
keyword	2161	5190/M	5353/M	5357/M	5360/M	5365/M	5368/M	5371/M	5383/M
keyword	2368	5351							
kind	12	5028/M	5415/M						
kind	2228	5491/M							
kind	5002	5011/M	5013/M	5015/M	5019				
kind	5327	5396/M	5398/M	5400/M	5406				
length	392	4684	4709/P	4710/P	4711/P	4714			
length	3843	3850	3853/M	3853					
length	4578	4655	4655/M	4655	4709	4709/M	4709		
list_overflow	4597	4668/M	4671	4671	4672/P				
list_overflow	5773	5793/M							
list_p	4598	4638	4639/P	4642	4645	4646	4648	4653	4654
		4695	4695	4720					
local_status	4599	4719/P							
local_status	5328	5441/P							
max_length	2170	5361/M	5392/M						
max_length	2372	5361							
max_length_index	5329	5363/M	5392/S						
max_length_specified	5330	5344/M	5362/M	5391					
media	1161	4668							
memory_list_index	4600	4686/M	4687	4687	4688/M	4695	4695/S	4695	4695/S
		4696/M	4696	4702					
mmc$	1509	1515	1518	1521	1524	1527	1530	1533	1537
		1540	1543	1546	1549	1552	1555	1559	1562
		1565	1568	1571	1574	1578	1581	1584	1587
		1590	1593	1596	1599	1602	1605	1608	1611
		1614	1617	1620	1623	1626	1629	1632	1635
		1638	1642	1645	1648	1651	1654	1657	1660
		1663	1666	1669	1672	1675	1678	1681	1684
		1687	1690	1693	1696	1699	1702	1706	1710
		1713	1717	1720	1723	1726	1729	1732	1735
		1738	1741	1744	1747	1750	1753	1756	1759
		1762	1765	1768	1771	1774	1778	1782	1785
		1788	1791	1794	1797	1800	1803	1806	1809

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		1812	1815	1818	1821	1824	1827	1830	1833
		1836	1840	1843	1846				
mmc$as_random	2133	4621							
mmc$as_read_tu	2134	4619							
mmc$as_sequential	2133	4617	4626/P	5025	5026/P	5412	5413/P		
mmc$assign_active_null	2972	2973							
mmc$cell_pointer	2224	2229	4630/P	5011	5396	5491			
mmc$default_sdt_length	3737	3730	4558						
mmc$heap_pointer	2225	2233	5015	5400					
mmc$kw_asid	2150	2186							
mmc$kw_clear_space	2148	2173							
mmc$kw_current_segment_length	2147	2167							
mmc$kw_error_exit_procedure	2149	2177							
mmc$kw_gl_key	2149	2171							
mmc$kw_hardware_attributes	2151	2180							
mmc$kw_inheritance	2151	2188							
mmc$kw_max_segment_length	2148	2169	5360						
mmc$kw_null_keyword	2146	5383							
mmc$kw_preset_value	2150	2175	5365						
mmc$kw_ps_transfer_size	2152	2196							
mmc$kw_ring_numbers	2146	2162	5353						
mmc$kw_segment_access_control	2150	2184	5368						
mmc$kw_segment_number	2147	2165	5357						
mmc$kw_shadow_segment	2152	2190	5190						
mmc$kw_software_attributes	2149	2182							
mmc$kw_wired_segment	2152	2193	5371						
mmc$sa_free_behind	2212	4616							
mmc$sa_read_transfer_unit	2212	4616	4618						
mmc$sas_terminate_access	3011	6172	6498						
mmc$segment_fault_processor_id	3441	3495							
mmc$sequence_pointer	2224	2231	5013	5398					
mmc$ssk_none	3051	3028							
mmc$ssk_read_only_file	3051	5890/P							
mmc$ssk_read_write_file	3051	5937/P							
mmc$ssk_segment_number	3052	3026							
mmc$ua_max_segment_length	2363	2371	5359						
mmc$ua_null_keyword	2365	5382							
mmc$ua_preset_value	2363	2373	5364						
mmc$ua_ring_numbers	2365	2380	5352						
mmc$ua_segment_access_control	2364	2375	5367						
mmc$ua_segment_number	2362	2369	5356						
mmc$ua_wired_segment	2364	2377	5370						
mmc$va_binding	2391	6535	6566/P						
mmc$va_execute	2390	6529	6533/P	6590/P					
mmc$va_pointer_to_procedure	2390	6537							
mmc$va_read	2389	6522	6534/P	6550/P	6567/P				
mmc$va_read_execute	2390	6532							
mmc$va_read_write	2389	6526							
mmc$va_write	2389	6524							
mme$volume_unavailable	1778	6615/P	6616/P						
mme$wired_seg_length_too_large	1785	5373/P							
mmk$close	1889	4893	4903						
mmk$create_scratch_segment	1967	5008	5039						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmk$create_segment	1922	5111	5130					
mmk$create_shadow_segment	2011	5179	5200					
mmk$delete_scratch_segment	1971	5489	5502					
mmk$delete_segment	1918	5546	5556					
mmk$fetch_seg_attributes	1900	5751	5764					
mmk$get_segment_length	1955	5838	5849					
mmk$initiate_shadowing	1987	5922	5941					
mmk$job_base	2053	1854	1858	1862	1866	1870	1874	1878	1882
		1886	1889	1893	1897	1900	1904	1907	1910
		1914	1918	1922	1926	1929	1932	1935	1938
		1941	1944	1947	1951	1955	1959	1963	1967
		1971	1975	1979	1983	1987	1991	1995	1999
		2003	2007	2011	2015	2019	2023		
mmk$open_file_segment	1893	6015	6033						
mmk$set_access_selections	1963	6229	6239						
mmk$set_segment_length	1959	6283	6292						
mmk$store_segment_attributes	1910	6353	6364						
mmk$terminate_shadowing	1991	6399	6412						
mmk$verify_access	1914	6483	6606						
mmp$advise_out	3859	4656	4710	4711					
mmp$assign_contiguous_memory	3866	5434							
mmp$build_segment	3874	5022	5124	5194	5252	5409	6027		
mmp$change_seg_inheritance_r1	3883	4755							
mmp$change_segment_inheritance	4742	4760							
mmp$change_segment_number	4794	4820							
mmp$change_segment_number_r1	3892	4810							
mmp$change_stack_attribute	4838	4854							
mmp$change_stack_attribute_r1	3902	4849							
mmp$close_segment	4882	4719	4905						
mmp$close_shared_stack	4924	4942							
mmp$convert_ps_transfer_size	3912	3937							
mmp$create_scratch_segment	4994	5041							
mmp$create_segment	5095	5132							
mmp$create_shadow_segment	5161	5202							
mmp$create_shared_stack	5227	5259							
mmp$create_user_segment	5312	5459							
mmp$delete_scratch_segment	5479	5504							
mmp$delete_segment	5535	5558							
mmp$delete_user_segment	5581	5441	5599						
mmp$failed_allocation_flag_hdl	5618	5688							
mmp$fetch_offset_mod_pages_r1	3949	5706							
mmp$fetch_offset_modified_pages	5691	4639	5715						
mmp$fetch_segment_attributes	5740	5765							
mmp$fetch_segment_attributes_r1	3960	5757							
mmp$get_allocated_addresses	5768	5796							
mmp$get_allocated_addresses_r1	3969	5787							
mmp$get_page_size	3980	3985	5448						
mmp$get_sdt_entry_p	4050	6168	6492						
mmp$get_sdt_entry_p	4052	4054/M	6168/M	6492/M					
mmp$get_sdtx_entry_p	4089	6169	6493						
mmp$get_sdtx_entry_p	4091	4093/M	6169/M	6493/M					
mmp$get_segment_length	5826	5851							
mmp$get_segment_length_r1	4041	5843							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmp$initiate_debug_shadowing	5867	5895						
mmp$initiate_shadowing	5913	5943						
mmp$initiate_shadowing_r1	4102	5890	5937					
mmp$invalidate_segment	4111	4896	4935	5494	5549	5591		
mmp$open_file_segment	5998	4630	6035					
mmp$open_segment	6038	6052						
mmp$preset_page_streaming	6075	6102						
mmp$preset_page_streaming_r1	4120	6093						
mmp$process_file_alloc	4132	5639						
mmp$reserve_segment_number	6105	6132						
mmp$reserve_segment_number_r1	4139	6123						
mmp$reverify_access	6156	6173						
mmp$reverify_access	6157	6170/M						
mmp$set_access_selections	6219	4626	4718	5026	5413	6241		
mmp$set_access_selections_r1	4148	6234						
mmp$set_segment_length	6271	6294						
mmp$set_segment_length_r1	4157	6288						
mmp$store_segment_attributes	6342	6366						
mmp$store_segment_attributes_r1	4166	6359						
mmp$terminate_shadowing	6389	6414						
mmp$terminate_shadowing_r1	4175	6408						
mmp$validate_segment_number	4183	4610						
mmp$verify_access	6434	6533	6534	6550	6566	6566	6590	6608
mmp$verify_access	6436	6479/M	6489/M	6495/M	6499/M	6507/M	6523/M	6525/M 6527/M
		6530/M	6533/M	6536/M	6551/M	6556/M	6561/M	6568/M 6573/M
		6591/M	6597/M	6602/M				
mmp$volume_unavailable_flag_hdl	6612	6618						
mmp$write_modified_pages	4192	5882	5928					
mmt$access_selections	2133	4151	4586	4996	5315	6221	6395	
mmt$ast_index	1303	1148	2883					
mmt$attribute_descriptor	2160	2347	3780	3963	4169	5096	5228	5335 5742
		5749	6000	6040	6345	6351		
mmt$attribute_keyword	2146	2161						
mmt$eoi_state	1312	1150						
mmt$hardware_attribute_set	2215	2181						
mmt$hardware_attributes	2203	2215						
mmt$lock_segment_status	2337	2937						
mmt$max_sdt	2893	2897						
mmt$max_sdtx	2961	2965						
mmt$sdtx_stream_data	2944	2940						
mmt$segment_access_condition	3468	3496						
mmt$segment_access_rights	3000	2936						
mmt$segment_access_state	3006	2931						
mmt$segment_attrib_descriptor	2342	3875	5005	5107	5176	5238	5336	6011
mmt$segment_descriptor	2880	2890	2894	3941	4052	4185	4602	6160 6473
mmt$segment_descriptor_extended	2929	2958	2962	3942	4091	4094	4186	4603 6161
		6169	6474	6493				
mmt$segment_inheritance	2242	2188	2933	3886	4744			
mmt$segment_pointer	2227	3782	3877	4589	4883	4925	5003	5099 5105
		5166	5174	5231	5236	5337	5485	5536 6004
		6009	6042	6396				
mmt$segment_pointer_kind	2224	2228	2345	3781	5002	5097	5165	5229 5327
		6001	6041					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmt$segment_reservation_state	3041	2934						
mmt$shadow_info	3021	2938						
mmt$shadow_reference_info	3055	2533						
mmt$shadow_segment_kind	3051	3025	4105					
mmt$software_attribute_set	2217	2183	2935	4615	4618			
mmt$software_attributes	2211	2217						
mmt$user_attribute_descriptor	2367	5313						
mmt$user_attribute_keyword	2362	2368						
mmt$va_access_mode	2389	6436						
mmt$xcb_page_wait_info	3066	2519						
mmv$file_allocation_interval	4533	5638						
mtt$monitor_interlock	1319	1138						
nat$received_message_descriptor	3082	3075	3084					
nat$received_message_list	3074	2501						
new_allocated_length	5623	5639/P	5643	5643	5646			
new_segment_pointer	4798	4813/M	4814/M					
nlc$cc_connect_confirm	3114	3105						
nlc$cc_connect_request	3113	3103						
nlc$cc_expedited_data	3119	3105						
nlc$cc_max_pdu_kind	3121	3124						
nlc$channel_connection_pdu	3137	3089						
nlc$channelnet_pdu	3137	3091						
nlt$cc_pdu_kind	3124	3102						
nlt$cc_seq#_or_connect_time	3101	3090						
nlt$cc_sequence_number	3127	3106						
nlt$device_identifier	3134	3085						
nlt$pdu_type	3137	3088						
normal	2290	4226/M	4611	4632	4641	4674	4752/M	4756 4757/M
		4807/M	4812	4817/M	4846/M	4850	4851/M	4894/M 4898
		4899/M	4934/M	4936	4937/M	5009/M	5024	5037/M 5112/M
		5125	5126/M	5180/M	5195	5196/M	5241/M	5253 5254/M
		5343/M	5411	5440	5456/M	5490/M	5496	5499/M 5547/M
		5551	5554/M	5590/M	5593	5596/M	5640	5642 5704/M
		5708	5709/M	5753/M	5758	5759/M	5784/M	5789 5790/M
		5839/M	5844	5845/M	5877/M	5883	5891	5892/M 5923/M
		5929	5938	5939/M	6016/M	6028	6029/M	6091/M 6095
		6096/M	6117/M	6124	6125/M	6231/M	6235	6236/M 6285/M
		6289	6290/M	6355/M	6360	6361/M	6400/M	6409 6410/M
offset_list	4583	4598						
offset_list	5694	5705	5711/M					
offsets_returned	4601	4636/M	4638	4640/P	4645	4648	4650	4652 4688
offsets_returned	5695	5712/M						
osc$base_exception	469	475						
osc$binding_segment	2283	6536						
osc$call_instruction	3332	3340						
osc$data_read	3331	3340						
osc$free_running_clock_maximum	1437	1434						
osc$invalid_ring	407	447						
osc$max_fault_contents	3508	3502						
osc$max_name_size	2840	2844	2847					
osc$max_page_size	1425	1421						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
osc$max_ring	406	447	448					
osc$max_segment_length	430	453	2941	2972				
osc$max_status_condition_code	2305	2301	2317					
osc$max_string_size	2321	2324	2327	2332	4211			
osc$max_tasks	1265	1262						
osc$maximum_offset	429	430	450	450	451	5882/P	5928/P	
osc$maximum_processor_id	3380	3376						
osc$maximum_segment	428	449						
osc$min_ecc	468	469						
osc$min_page_size	1424	1421						
osc$min_ring	405	448						
osc$non_executable	2279	6531						
osc$non_readable	2282	6523	6528					
osc$non_writable	2285	6525	6527					
osc$pr_base_constant	4003	4312	5630	6167	6487			
osc$pr_page_size_mask	4006	3983	5448					
osc$segnum_job_fixed_heap	4068	4311	5630	6167	6487			
osc$task_time_slice_maximum	2705	2708						
osc$vl_invalid_entry	2908	6171	6494					
osc$wait	4205	5882/P	5928/P					
osk$base	2058	2395	2399	2403	2407	2411	2415	2419	2423
		2427	2431	2435	2439	2443	2447	2452	2455
		2458							
osk$entry	2090	4893	5008	5111	5179	5489	5546	5751	5838
		5922	6015	6229	6283	6399	6483		
osk$exit	2091	4903	5039	5130	5200	5502	5556	5764	5849
		5941	6033	6239	6292	6353	6364	6412	6606
osk$m	2129	4893	5179	5489	5546	5751	5838	5922	6033
		6229	6292	6399	6483				
osk$system_class	2104	2088	2089	2090	2091	2092	2093	2094	
osp$set_status_abnormal	4208	5373							
osp$set_status_condition	4221	4229	4757	4817	4851	4899	4937	5037	5126
		5196	5254	5456	5499	5554	5596	5709	5759
		5790	5845	5892	5939	6029	6096	6125	6236
		6290	6361	6410					
osp$verify_system_privilege	4243	4285	6624						
osp$wait_for_unavailable_volume	6621	6616	6628						
osp$wait_on_condition	4287	5668	6626						
ost$asid	2260	2187	2256	2914					
ost$binary_unique_name	1333	1140							
ost$byte_count	2250	3860	4193						
ost$caller_identifier	2469	4262	4748	4802	4843	4888	4930	5001	5104
		5170	5235	5321	5484	5541	5586	5746	5777
		5833	5872	5918	6008	6047	6084	6225	6278
		6349	6451						
ost$cp_time	3170	2518							
ost$cp_time_value	3168	2531	3171	3172					
ost$cs_lock	1444	2499							
ost$debug_code	3331	3319							
ost$debug_list	3327	3231							
ost$debug_list_entry	3318	3327							
ost$debug_mask	3337	3230							
ost$exchange_package	3180	2486							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
ost$execute_privilege	2279	2274	2909					
ost$execution_control_block	2485	2511	4051	4090	4299	5628	6163	6477
ost$flags	3237	3187						
ost$frame_descriptor	3295	3310						
ost$free_running_clock	1434	1156	2517	2657				
ost$global_task_id	1256	1158	1249	2496	2497	3392	3595	
ost$heap	4085	4546						
ost$key_lock	436	2172	2915					
ost$key_lock_value	442	439	2471	2473	3254	3256		
ost$keypoint_class	3269	3200	3271					
ost$keypoint_mask	3271	3203						
ost$minimum_save_area	3305	3192	3280	3489				
ost$monitor_condition	3141	3148						
ost$monitor_conditions	3148	3193	3197	3285	3564	3578		
ost$monitor_fault	3485	3434						
ost$monitor_fault_contents	3502	3498						
ost$name	2847	2620	2836	3527	3788	6039		
ost$p_register	3252	3181	3306	3556	3562			
ost$page_size	1421	1402	4540					
ost$paging_statistics	3358	2526						
ost$processor_id	3376	2489	3370					
ost$processor_id_set	3370	2488						
ost$processor_model_number	1351	1335						
ost$processor_serial_number	1429	1334						
ost$pva	458	3225	3243	3257	3486	3579	6445	6447	6452
		6456	6459						
ost$read_privilege	2282	2275	2910						
ost$register_number	3248	3222	3291	3299	3300	3301			
ost$ring	447	459	2163	2164	2381	2382	2474	2912	2913
		2930	3242	4564	6472				
ost$ring1_termination_reason	3388	2522							
ost$segment	449	460	1157	2166	2370	2475	3027	3220	3321
		3884	3893	3894	3940	3950	3961	3970	4042
		4052	4091	4112	4121	4141	4149	4158	4167
		4175	4184	4604	4796	5177	5625	6107	6112
		6162	6475						
ost$segment_access_control	2272	2185	2376						
ost$segment_descriptor	2907	2881							
ost$segment_length	453	392	2168	2170	2192	2194	2197	2372	2378
		3868	4044	4160	4591	5164	5829	5834	6274
ost$segment_offset	450	391	461	2257	2945	3322	3324	3756	3953
		3972	4583	4592	4606	5163	5694	5700	5770
ost$stack_frame_save_area	3279	3313	3522						
ost$status	2289	2179	3557	3750	3760	3783	3861	3869	3878
		3887	3896	3905	3944	3955	3964	3976	4045
		4106	4115	4127	4134	4142	4152	4161	4170
		4176	4187	4195	4212	4223	4293	4318	4580
		4599	4745	4749	4799	4804	4840	4844	4885
		4889	4927	4931	4998	5004	5100	5106	5167
		5175	5232	5237	5317	5328	5333	5481	5486
		5538	5542	5583	5587	5626	5696	5702	5743
		5748	5774	5782	5830	5835	5869	5874	5915
		5920	6005	6010	6043	6081	6087	6108	6113

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		6222	6226	6275	6279	6346	6350	6392	6397
		6476							
ost$status_condition_code	2317	2292	2313	3743	4210	4222	4288	6622	
ost$status_identifier	4217	4209							
ost$string	2330	2293							
ost$string_size	2324	2331							
ost$system_flag	3663	3659	5619	6613					
ost$system_privilege_map	3730	4255	4557	4557					
ost$task_index	1262	1257	1296	1297					
ost$task_time_slice	2708	2694							
ost$top_of_stack_pointer	3240	3232							
ost$trap_enable	3274	3189	3553						
ost$user_condition	3151	3158							
ost$user_conditions	3158	3191	3195	3283	3312	3525	3565		
ost$valid_relative_pointer	456	1163	1166	2515	2516				
ost$valid_ring	448	2343	3232	3779	3885	3895	3904	3962	3971
		4043	4104	4113	4122	4150	4159	4168	4565
		4566	4750	4797	4805	4884	4890	5098	5108
		5537	5543	5828	5836	5875	6002	6012	6227
		6273	6281	6344					
ost$virtual_machine_identifier	3262	3183	3185	3307					
ost$wait	4205	4194							
ost$write_privilege	2285	2276	2911						
ost$x_register	3249	3222	3291						
osv$page_size	4540	3925	3925	4655/P	4656/P	4698	4699		
osv$system_privilege_map	4255	4273							
osv$system_privilege_map	6621	6624							
osv$task_private_heap	4546	4638	4642	4646	4720				
p	4263	4277/M	4281/M						
p	5003	5022/P	5026/P	5030	5032	5034			
p	5485	5491/M	5492/M						
p	6621	6624/M	6624/M						
page_size	3980	3983/M							
page_size	5312	5448/M							
page_size	5331	5448/P	5452						
pmc$account_log	4332	4340	4342						
pmc$external_log_base_exception	4375	4378	4381	4385	4388	4392	4395	4398	4401
		4405	4409	4412	4416	4419	4423	4427	4430
		4434	4438	4441	4444	4447	4450	4454	4457
		4460	4463	4466	4469	4472	4475	4478	4481
		4484	4487	4490	4493	4496	4499	4502	4505
		4508							
pmc$job_account_log	4332	4338	4344						
pmc$job_log	4334	4336							
pmc$job_statistic_log	4332	4344							
pmc$kill_task_flag	3663	3679	5671						
pmc$max_signal_contents	3646	3640							
pmc$max_task_id	3401	3398							
pmc$min_ecc	4365	4361							
pmc$pc_base_exception	4361	4375							
pmc$program_aborting	4519	4521							
pmc$program_exiting	4518	4521							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pmc$sf_terminate_task	3664	5670							
pmc$statistic_log	4333	4338	4342						
pmc$system_log	4334	4336	4340						
pmc$task_terminating	4519	5634							
pmp$exit	4293	5674							
pmp$find_executing_task_xcb	4298	4313	5630						
pmp$log	4317	5673							
pmp$task_state	4514	5634							
pmt$ascii_logs	4336	4348							
pmt$binary_logs	4338	4350							
pmt$condition_identifier	3475	3469							
pmt$cpu_model_number	1411	1400	1407						
pmt$cpu_serial_number	1414	1401	1406						
pmt$global_binary_logs	4342	4354							
pmt$global_logs	4340	4352							
pmt$initialization_value	1488	1155	2176	2374	5172	5173			
pmt$local_binary_logs	4344	4356							
pmt$log_msg_text	4326	4317							
pmt$logs	4332	4346							
pmt$signal	3602	3596							
pmt$signal_contents	3640	3604							
pmt$signal_id	3607	3603							
pmt$task_id	3398	2513	3393	3876	4114	4926	5230		
pmt$task_state	4518	4514							
pointer	4883	4893	4896/P	4901/M					
pointer	4925	4935/P	4939/M						
pointer	4997	5028/M	5030/M	5032/M	5034/M				
pointer	5099	5128/M							
pointer	5231	5256/M							
pointer	5316	5415/M	5420/M	5425/M	5430/M	5434/P	5441/P		
pointer	5480	5489	5492	5494/P	5497/M				
pointer	5536	5546	5549/P	5552/M					
pointer	5582	5591/P	5594/M						
pointer	6004	6031/M	6033						
pointer	6042	6050/P							
pointer	6453	6481/M	6482	6505	6518/M	6533/P	6534/P	6550/P	6555
		6560	6565/M	6566/P	6566/P	6572	6582	6595	6596
pointer_kind	2345	5019/M	5116/M	5188/M	5244/M	5406/M	6020/M		
pointer_kind	4995	5010	5012	5028	5029	5031			
pointer_kind	5097	5116							
pointer_kind	5165	5188							
pointer_kind	5229	5244							
pointer_kind	5314	5395	5397	5415	5416	5421			
pointer_kind	6001	6020							
pointer_kind	6041	6050/P							
power	3917	3926/M	3928/M	3928	3931	3934			
present	3587	5679							
preset_and_save_ts_fb	6076	6093/P							
preset_pointer	5332	5419	5424	5429	5450	5450	5451		
preset_value	2176	5366/M							
preset_value	2374	5366							
previous_allocated_length	5624	5631/M	5643	5646/M					
ps_transfer_size	3912	3925							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ps_transfer_size_power	3913	3931	3932/M	3932	3934/M				
ptr	6464	6480/M	6481						
pva	4743	4754/P	4755/P						
pva	5166	5198/M							
pva	5741	5751	5757/P						
pva	5827	5838	5843/P						
pva	6077	6093/P							
pva	6220	6229	6233/P	6234/P					
pva	6272	6287	6292						
pva	6343	6359/P							
pva	6454	6455							
pva	6456	6482	6505	6518/M	6533/P	6534/P	6550/P	6565/M	6566/P
		6566/P							
pva_p	6157	6165							
pva_p	6435	6480							
pva_p	6467	6480/M							
r1	2163	5354/M							
r1	2381	5354							
r1	2912	6525	6527	6530					
r1_addr_list_p	5779	5786	5787/P	5792					
r1_addr_returned	5780	5788/P	5794						
r1_free_behind	6085	6094/P	6099						
r1_list_overflow	5781	5788/P	5793						
r1_num_array_p	6112	6118	6120/M	6123/P	6128				
r1_offset_list_p	5700	5705	5707/P	5711					
r1_offsets_returned	5701	5707/P	5712						
r1_pointer	5105	5124/P	5128						
r1_pointer	5174	5194/P	5198						
r1_pointer	5236	5252/P	5256						
r1_pointer	5873	5889/M	5890/P						
r1_pointer	5919	5936/M	5937/P						
r1_pointer	6009	6027/P	6031						
r1_segment_attributes_p	5749	5755	5756/M	5757/P	5761				
r1_segment_attributes_p	6351	6357	6358/M	6359/P					
r1_segment_length	5834	5843/P	5847						
r1_status	4749	4755/P	4756	4757/P					
r1_status	4804	4811/P	4812	4817/P					
r1_status	4844	4849/P	4850	4851/P					
r1_status	4889	4897/P	4898	4899/P					
r1_status	4931	4935/P	4936	4937/P					
r1_status	5004	5022/P	5024	5037/P					
r1_status	5106	5124/P	5125	5126/P					
r1_status	5175	5194/P	5195	5196/P					
r1_status	5237	5252/P	5253	5254/P					
r1_status	5333	5409/P	5411	5456/P					
r1_status	5486	5495/P	5496	5499/P					
r1_status	5542	5550/P	5551	5554/P					
r1_status	5587	5592/P	5593	5596/P					
r1_status	5702	5707/P	5708	5709/P					
r1_status	5748	5757/P	5758	5759/P					
r1_status	5782	5788/P	5789	5790/P					
r1_status	5835	5843/P	5844	5845/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
r1_status	5874	5890/P	5891	5892/P					
r1_status	5920	5937/P	5938	5939/P					
r1_status	6010	6027/P	6028	6029/P					
r1_status	6087	6094/P	6095	6096/P					
r1_status	6113	6123/P	6124	6125/P					
r1_status	6226	6234/P	6235	6236/P					
r1_status	6279	6288/P	6289	6290/P					
r1_status	6350	6359/P	6360	6361/P					
r1_status	6397	6408/P	6409	6410/P					
r1_transfer_size	6086	6094/P	6098						
r2	2164	5355/M							
r2	2382	5355							
r2	2913	6523	6530	6583	6584	6585			
r3	6444	6572							
ref_r	6472	6505/M	6506	6516	6517/M	6523	6525	6527	6530
		6530	6565	6572	6583	6585/M	6587		
residence	1496	3808	3809	4667	4667				
return_unallocated_offsets	5693	5706/P							
ring	459	6505	6518/M	6555	6560	6565/M	6584/M	6587/M	6596
ring	2474	4754/P	4809/P	4849/P	4895/P	4935/P	5017	5113/P	5186
		5242	5404	5494/P	5548/P	5591/P	5757/P	5787/P	5842/P
		5890/P	6017/P	6050/P	6093/P	6233/P	6286/P	6359/P	6516
		6517	6518						
rp	2910	6523	6528	6536					
s	6461	6481/M							
s_p	6469	6481							
saved_free_behind	6080	6099/M							
saved_transfer_size	6079	6098/M							
sd_p	6473	6492/M	6494	6523	6523	6525	6525	6527	6527
		6528	6530	6530	6531	6536	6583	6584	6585
sdt_entry_p	6160	6168/M	6171						
sdt_offset	2515	4055	6168	6492					
sdt_p	4602	4610/P							
sdtx_entry_p	6161	6169/M	6171						
sdtx_offset	2516	4094	6169	6493					
sdtx_p	4603	4610/P	4615	4618	4630/P	4667/P	4672/P		
sdtx_p	6474	6493/M	6498						
seg	460	6482							
seg	3805	3807/M	3810/M	3812/M	3815	3816			
seg	4578	4667/M	4667/M	4667/M	4667	4667			
seg_attrib_p	5335	5348	5353/M	5354/M	5355/M	5357/M	5358/M	5360/M	5361/M
		5365/M	5366/M	5368/M	5369/M	5371/M	5376/M	5383/M	5388/M
		5392/M	5392	5407	5434/P				
seg_attributes	5742	5755	5755	5756	5761/M				
seg_attributes	6040	6050/P							
seg_attributes_p	5096	5117	5120	5121	5122				
seg_attributes_p	5228	5245	5247	5248	5249				
segment_attributes	5005	5017/M	5018/M	5019/M	5020/M	5021/M	5022/P		
segment_attributes	5107	5114/M	5115/M	5116/M	5117/M	5118/M	5119	5120	5122/M
		5124/P							
segment_attributes	5176	5186/M	5187/M	5188/M	5189	5190/M	5191/M	5192/M	5193/M
		5194/P							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
segment_attributes	5238	5242/M	5243/M	5244/M	5245/M	5246	5247	5249/M	5251/M
		5252/P							
segment_attributes	5336	5404/M	5405/M	5406/M	5407/M	5408/M	5409/P		
segment_attributes	6011	6018/M	6019/M	6020/M	6021/M	6022	6023	6024/M	6026/M
		6027/P							
segment_inheritance	4744	4755/P							
segment_length	5829	5840/M	5847/M						
segment_length	6274	6288/P							
segment_num_array	6107	6118	6118	6119	6119	6127	6127	6128/M	
segment_number	4796	4810/P	4815						
segment_p	4579	4610/P	4626/P	4639/P	4653	4707	4718/P		
segment_p	5162	5179	5182	5182					
segment_p	5692	5706/P							
segment_p	5914	5922	5928/P	5936	5937/P				
segment_p	6390	6399	6405/P	6408/P					
segment_pointer	4795	4810/P	4813	4814	4815				
segment_pointer	5337	5409/P	5413/P	5417	5419	5420	5422	5424	5425
		5427	5429	5430					
segment_pointer	5868	5882/P	5889						
segment_pointer	6280	6287/M	6288/P						
segment_table_length	3220	6170	6488						
segnum	2166	5358/M							
segnum	2370	5358							
segnum	2475	4273/S	6624/S						
segnum	4052	4055							
segnum	4091	4094							
segnum	6155	6168							
segnum	6156	6169							
segnum	6162	6165/M	6168/P	6169/P	6170				
segnum	6434	6492							
segnum	6434	6493							
segnum	6475	6482/M	6483	6488	6492/P	6493/P			
seq_p	5338	5417/M	5418	5419	5422/M	5423	5424	5427/M	5428
		5429							
seq_pointer	2232	5032	5422	5424	5425				
sequence_pointer	18	5032/M	5425/M						
sfc$no_limit	2357	4630/P	5115	5187	5243				
sfc$temp_file_space_limit	2358	5018	5405						
sfid	2346	5021/M	5118/M	5193/M	5251/M	5408/M	6026/M		
sfid	2932	4667/P	4672/P						
sfid	3799	3808	3809	3815	3816				
sfid	4578	4667	4667	4667	4667				
sfid	5999	6026							
sft$file_space_limit_kind	2357	2344	2939	4594	5324	6003			
shadow_info	2938	4630/P							
shadow_length	2192	5192/M							
shadow_length	5164	5192							
shadow_offset	5163	5182							
shadow_p	2191	5191/M							
shadow_sfid	3024	4630/P							
shared_stack_flag	6106	6123/P							
shared_taskid_array	4926	4935/P							
shared_taskid_array	5230	5252/P							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
signals	2525	5679							
size	2331	4228/M	4757/M	4817/M	4851/M	4899/M	4937/M	5037/M	5126/M
		5196/M	5254/M	5456/M	5499/M	5554/M	5596/M	5709/M	5759/M
		5790/M	5845/M	5892/M	5939/M	6029/M	6096/M	6125/M	6236/M
		6290/M	6361/M	6410/M					
software_attribute_set	2935	4615	4618						
source	3841	3851							
source	4578	4655	4709						
source	4605	4653/M	4655/P	4707/M	4709/P	4710/P			
stack_pages_to_be_freed	4839	4849/P							
starting_addr	4606	4666/M	4672/P	4714/M					
starting_byte_address	5770	5787/P							
static_link	6459	6555							
status	4223	4226/M	4227/M	4228/M					
status	4580	4610/P	4611	4626/P	4631/P	4632	4640/P	4641	4656/P
		4673/P	4674	4710/P	4711/P	4718/P			
status	4742	4757/M	4757/M	4757/M					
status	4745	4752/M	4757/P						
status	4794	4817/M	4817/M	4817/M					
status	4799	4807/M	4817/P						
status	4838	4851/M	4851/M	4851/M					
status	4840	4846/M	4851/P						
status	4882	4899/M	4899/M	4899/M					
status	4885	4894/M	4899/P						
status	4924	4937/M	4937/M	4937/M					
status	4927	4934/M	4937/P						
status	4994	5037/M	5037/M	5037/M					
status	4998	5009/M	5026/P	5037/P					
status	5095	5126/M	5126/M	5126/M					
status	5100	5112/M	5126/P						
status	5161	5196/M	5196/M	5196/M					
status	5167	5180/M	5196/P						
status	5227	5254/M	5254/M	5254/M					
status	5232	5241/M	5254/P						
status	5312	5456/M	5456/M	5456/M					
status	5317	5343/M	5373/P	5413/P	5435/P	5440	5456/P		
status	5479	5499/M	5499/M	5499/M					
status	5481	5490/M	5499/P						
status	5535	5554/M	5554/M	5554/M					
status	5538	5547/M	5554/P						
status	5581	5596/M	5596/M	5596/M					
status	5583	5590/M	5596/P						
status	5626	5639/P	5640	5642	5642	5655	5655	5656	5664/P
		5673/P	5674/P						
status	5691	5709/M	5709/M	5709/M					
status	5696	5704/M	5709/P						
status	5740	5759/M	5759/M	5759/M					
status	5743	5753/M	5759/P						
status	5768	5790/M	5790/M	5790/M					
status	5774	5784/M	5790/P						
status	5826	5845/M	5845/M	5845/M					
status	5830	5839/M	5845/P						
status	5867	5892/M	5892/M	5892/M					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
status	5869	5877/M	5882/P	5883	5892/P				
status	5913	5939/M	5939/M	5939/M					
status	5915	5923/M	5928/P	5929	5939/P				
status	5998	6029/M	6029/M	6029/M					
status	6005	6016/M	6029/P						
status	6043	6050/P							
status	6075	6096/M	6096/M	6096/M					
status	6081	6091/M	6096/P						
status	6105	6125/M	6125/M	6125/M					
status	6108	6117/M	6125/P						
status	6219	6236/M	6236/M	6236/M					
status	6222	6231/M	6236/P						
status	6271	6290/M	6290/M	6290/M					
status	6275	6285/M	6290/P						
status	6342	6361/M	6361/M	6361/M					
status	6346	6355/M	6361/P						
status	6389	6410/M	6410/M	6410/M					
status	6392	6400/M	6405/P	6410/P					
ste	2881	6171	6494	6523	6523	6525	6525	6527	6527
		6528	6530	6530	6531	6536	6583	6584	6585
str	6454	6460							
str1	3846	3851/M	3853						
str1	4578	4655/M	4655	4709/M	4709				
str2	3847	3852/M	3853/M	3854					
str2	4578	4655/M	4655/M	4655	4709/M	4709/M	4709		
syc$ucr_condition	3513	3524							
syc$user_defined_condition	3514	3526							
system_flags	2500	5670	5677						
system_table_lock_count	2499	5672							
syt$monitor_flag	3420	3405							
syt$monitor_flags	3405	2487							
temp_transfer_size	6078	6093/P							
text	2293	4228/M	4757/M	4817/M	4851/M	4899/M	4937/M	5037/M	5126/M
		5196/M	5254/M	5456/M	5499/M	5554/M	5596/M	5709/M	5759/M
		5790/M	5845/M	5892/M	5939/M	6029/M	6096/M	6125/M	6236/M
		6290/M	6361/M	6410/M					
tmc$broken_task_fault_id	3441	3491							
tmc$btc_invalid_a0	3539	3560							
tmc$btc_invalid_p	3539	3560							
tmc$btc_mcr_traps_disabled	3540	3561							
tmc$btc_mf_traps_disabled	3539	3559							
tmc$btc_mntr_fault_buffer_full	3538	3559							
tmc$btc_system_error	3541	3555							
tmc$btc_ucr_traps_disabled	3540	3561							
tmc$dummy_fault	3442	3497							
tmc$flag_available_31	3676	3680							
tmc$maximum_monitor_faults	3446	3437							
tmc$maximum_signals	3656	3653	5678						
tmc$maximum_system_task_id	3689	3692							
tmc$mcr_fault	3441	3493							
tmc$signal_available_63	3638	3649							
tmc$stid_null_task	3695	3692							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmt$broken_task_condition	3538	3554							
tmt$broken_task_monitor_fault	3552	3492							
tmt$mcr_faults	3577	3494							
tmt$monitor_fault_buffer	3431	2524							
tmt$monitor_fault_buffers	3437	3432	3433	3434					
tmt$monitor_fault_identifiers	3440	3490	3566						
tmt$signal	3594	3589							
tmt$signal_buffer	3586	2525							
tmt$signal_buffers	3653	3587	3588	3589					
tmt$system_flags	3659	2500	5670	5671	5677				
tmt$system_task_id	3692	2491							
tmt$task_queue_link	1295	1288							
ts	3918	3925/M	3927	3927	3929/M	3929			
update	6391	6404							
update_passive_with_active	4578	4722	6405						
user_attributes	2347	5020/M	5117/M	5119	5120	5122/M	5189	5190/M	5191/M
		5192/M	5245/M	5246	5247	5249/M	5407/M	6021/M	6022
		6023	6024/M						
user_attributes_p	5313	5347	5348	5348	5350	5350	5351	5354	5355
		5358	5361	5366	5369	5372	5376	5377	
validating_ring_num	4566	4569/M	4571/M						
validating_ring_num	4742	4754/M	4754/M						
validating_ring_num	4794	4809/M	4809/M						
validating_ring_num	4882	4895/M	4895/M						
validating_ring_num	5095	5113/M	5113/M						
validating_ring_num	5535	5548/M	5548/M						
validating_ring_num	5826	5842/M	5842/M						
validating_ring_num	5998	6017/M	6017/M						
validating_ring_num	6219	6233/M	6233/M						
validating_ring_num	6271	6286/M	6286/M						
validating_ring_number	2343	5017/M	5114/M	5186/M	5242/M	5404/M	6018/M		
validating_ring_number	4750	4754/P	4755/P						
validating_ring_number	4805	4809/P	4811/P						
validating_ring_number	4890	4895/P	4896/P						
validating_ring_number	5108	5113/P	5114						
validating_ring_number	5543	5548/P	5549/P						
validating_ring_number	5836	5842/P	5843/P						
validating_ring_number	6012	6017/P	6018						
validating_ring_number	6227	6233/P	6234/P						
validating_ring_number	6281	6286/P	6288/P						
validation_ring_number	4565	4568	4571						
validation_ring_number	4742	4754	4754						
validation_ring_number	4794	4809	4809						
validation_ring_number	4797	4809/P							
validation_ring_number	4882	4895	4895						
validation_ring_number	4884	4895/P							
validation_ring_number	5095	5113	5113						
validation_ring_number	5098	5113/P							
validation_ring_number	5535	5548	5548						
validation_ring_number	5537	5548/P							
validation_ring_number	5826	5842	5842						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
validation_ring_number	5828	5842/P						
validation_ring_number	5998	6017	6017					
validation_ring_number	6002	6017/P						
validation_ring_number	6219	6233	6233					
validation_ring_number	6271	6286	6286					
validation_ring_number	6273	6286/P						
verify_access	6485	6485	6490	6496	6500	6508	6552	6557	6562
		6569	6574	6592	6598	6604			
vl	2908	6171	6494						
wait_time	5627	5638/M	5644/M						
wired_flag	5339	5345/M	5380/M	5391	5444				
wired_index	5340	5381/M	5392/S	5434/S					
wired_segment_length	2194	5376/M	5392	5434/P					
wired_segment_length	2378	5372	5376						
wp	2911	6525	6527						
xcb	4298	4311/M							
xcb	5618	5630/M							
xcb_p	4051	4054	4055						
xcb_p	4090	4093	4094						
xcb_p	5628	5630/P	5670	5672	5677	5679			
xcb_p	6156	6168	6168						
xcb_p	6156	6169	6169						
xcb_p	6163	6167/M	6168/P	6169/P	6170				
xcb_p	6434	6492	6492						
xcb_p	6434	6493	6493						
xcb_p	6477	6487/M	6488	6492/P	6493/P				
xp	2486	6170	6488						
xp	2909	6531							
xp	6442	6595							

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

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE

```
            3 MODULE mmm$segment_manager_system_core;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
o  5350
o  5351   PROCEDURE [XREF] dfp$set_server_eoi
o  5352     (     sfid: gft$system_file_identifier;
o  5353           segment_length: ost$segment_length;
o  5354      VAR status: ost$status);
o  5355
   5358
   5359   PROCEDURE [XREF] dmp$allocate_file_space_r1 ALIAS 'dmxaspa' (system_file_id:
   5360     dmt$system_file_id;
   5361         byte_address: amt$file_byte_address;
   5362         number_bytes_to_allocate: amt$file_byte_address;
   5363         chapter_number: dmt$chapter_number;
   5364         wait_option: ost$wait;
   5365         file_space_limit: sft$file_space_limit_kind;
   5366     VAR status: ost$status);
o  5384
o  5385   PROCEDURE [XREF] dmp$create_disk_file
o  5386     (     fde_p: gft$file_desc_entry_p;
o  5387           file_attributes_p: ^array [ * ] of dmt$file_attribute;
o  5388           allocation_length: amt$file_byte_address;
o  5389           sfid: gft$system_file_identifier;
o  5390      VAR status: ost$status);
o  5391
   5394
   5395   PROCEDURE [XREF] dmp$destroy_file ALIAS 'dmxfdes' (VAR system_file_id:
   5396     dmt$system_file_id;
   5397         file_space_limit: sft$file_space_limit_kind;
   5398     VAR status: ost$status);
o  5401 PROCEDURE [XREF] dmp$fetch_eoi ALIAS 'dmxfeoi' (system_file_id:
o  5402    dmt$system_file_id;
o  5403    VAR eoi:amt$file_byte_address;
o  5404    VAR status:ost$status);
   5407
   5408   PROCEDURE [INLINE] dmp$get_disk_file_descriptor_p (fde_p: gft$file_desc_entry_p;
   5409     VAR dfd_p: ^dmt$disk_file_descriptor);
   5410
o  5446
o  5447  PROCEDURE [XREF] dmp$get_initialized_addresses (sfid: dmt$system_file_id;
o  5448         starting_byte_address: ost$segment_offset;
o  5449    VAR addr_list: array [ * ] of dmt$addr_length_pair;
o  5450    VAR addr_returned: integer;
o  5451    VAR list_overflow: boolean;
o  5452    VAR status: ost$status);
o  5453
   5463
   5464
   5465     PROCEDURE [XREF] dmp$get_total_allocated_length
   5466       (     fde_p: gft$locked_file_desc_entry_p;
   5467        VAR allocated_length: amt$file_byte_address);
   5468
   5469
o  5472
o  5473  PROCEDURE [XREF] dmp$mfh_for_sfid (system_file_id: dmt$system_file_id;
o  5474         offset_requiring_allocation: amt$file_byte_address;
o  5475         file_space_limit: sft$file_space_limit_kind;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
o  5476     VAR status: ost$status);
o  5477
   5480 PROCEDURE [XREF] dmp$reallocate_file_space (sfid: dmt$system_file_id;
   5481         copy_pages: boolean;
   5482     VAR status: ost$status);
o  5485   PROCEDURE [XREF] dmp$sparse_allocate
o  5486     (     sfid: gft$system_file_identifier;
o  5487           offset_requiring_allocation: amt$file_byte_address;
o  5488           file_space_limit: sft$file_space_limit_kind;
o  5489      VAR status: ost$status);
o  5490
   5493
   5494
   5495      PROCEDURE [XREF] gfp$assign_fde
   5496        (     residence: gft$table_residence;
   5497              segment_number: ost$segment;
   5498         VAR sfid: gft$system_file_identifier;
   5499         VAR fde_p: gft$file_desc_entry_p);
   5500
o  5503
o  5504
o  5505    PROCEDURE [XREF] gfp$free_fde
o  5506      (     fde_p: gft$file_desc_entry_p);
o  5507
   5510
   5511   FUNCTION [UNSAFE, INLINE] gfp$get_eoi_from_fde
   5512     (      fde_p: gft$file_desc_entry_p): amt$file_byte_address;
   5513
o  5532
o  5533   PROCEDURE [INLINE] gfp$get_fde_p (sfid: gft$system_file_identifier;
o  5534     VAR fde_p: gft$file_desc_entry_p);
o  5535
o  5536
   5570
   5571
   5572   PROCEDURE [INLINE] gfp$lock_fde (fde_p: gft$file_desc_entry_p);
   5573
   5574
o  5705
o  5706   PROCEDURE [INLINE] gfp$unlock_fde_p (fde_p: gft$locked_file_desc_entry_p);
o  5707
o  5708
   5788
   5789   PROCEDURE [INLINE] gfp$get_locked_fde_p (sfid: gft$system_file_identifier;
   5790     VAR fde_p: gft$file_desc_entry_p);
   5791
   5792
o  5804   PROCEDURE [XREF] i#build_adaptable_heap_pointer (ring: 0 .. 15;
o  5805         segment: 0 .. 4095;
o  5806         offset: -80000000(16) .. 7fffffff(16);
o  5807         heap_length: 0 .. 7fffffff(16);
o  5808     VAR heap_p: ^HEAP ( * ));
o  5809
o  5810   PROCEDURE [XREF] i#build_adaptable_seg_pointer (ring: 0 .. 15;
o  5811         segment: 0 .. 4095;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
o  5812          offset: -80000000(16) .. 7fffffff(16);
o  5813          sequence_length: 0 .. 80000000(16);
o  5814            next_entry: 0 .. 7fffffff(16);
o  5815      VAR seq_p: ^SEQ ( * ));
o  5816
o  5817
o  5818    PROCEDURE [INLINE] i#move (source: ^cell;
o  5819          dest: ^cell;
o  5820          length: 0 .. 7fffffff(16));
o  5821
o  5822      VAR
o  5823        str1: ^string (65535),
o  5824        str2: ^string (65535);
o  5825
o  5827      IF length <> 0 THEN
o  5828        str1 := source;
o  5829        str2 := dest;
o  5830        str2^ (1, length) := str1^ (1, length);
o  5831        #SPOIL (str2^);
o  5832      IFEND;
o  5834    PROCEND i#move;
o  5835
o  5836    PROCEDURE [XREF] i#real_memory_address (p: ^cell;
o  5837      VAR rma: integer);
o  5838
o  5839    PROCEDURE [XREF] mmp$advise_in (pva: ^cell;
o  5840          length: ost$byte_count;
o  5841      VAR status: ost$status);
o  5842
   5845
   5846    PROCEDURE [XREF] mmp$advise_out (pva: ^cell;
   5847          length: ost$byte_count;
   5848      VAR status: ost$status);
   5849
o  5852
o  5853    PROCEDURE [XREF] mmp$assign_contiguous_memory
o  5854      (     process_virtual_address: ^cell;
o  5855            contiguous_mem_length: ost$segment_length;
o  5856       VAR status: ost$status);
o  5857
   5860
   5861  { Convert page streaming transfer size from bytes number of pages expressed as a power of 2
   5862
   5863    PROCEDURE [INLINE] mmp$convert_ps_transfer_size (ps_transfer_size: integer;
   5864          VAR ps_transfer_size_power: 0..15);
o  5890
o  5891    PROCEDURE [XREF] mmp$free_pages (pva: ^cell;
o  5892          length: ost$byte_count;
o  5893          waitopt: ost$wait;
o  5894      VAR status: ost$status);
o  5895
   5898
   5899    PROCEDURE [INLINE] mmp$get_max_sdt_sdtx_pointer
   5900      (     xcb_p: ^ost$execution_control_block;
   5901       VAR sdt_p: mmt$max_sdt_p;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
   5902       VAR sdtx_p: mmt$max_sdtx_p);
   5903
o  5912
o  5913     FUNCTION [INLINE] mmp$get_sdt_entry_p
o  5914       (     xcb_p: ^ost$execution_control_block;
o  5915             segnum: ost$segment): ^mmt$segment_descriptor;
o  5916
o  5917       mmp$get_sdt_entry_p := #address (1, #segment (xcb_p),
o  5918         8 * segnum + xcb_p^.sdt_offset);
o  5919
o  5920     FUNCEND;
o  5921
   5924
   5925     FUNCTION [INLINE] mmp$get_sdtx_entry_p
   5926       (     xcb_p: ^ost$execution_control_block;
   5927             segnum: ost$segment): ^mmt$segment_descriptor_extended;
   5928
   5929       mmp$get_sdtx_entry_p := #address (1, #segment (xcb_p),
   5930         #SIZE (mmt$segment_descriptor_extended) * segnum + xcb_p^.sdtx_offset);
   5931
   5932     FUNCEND;
   5933
o  5936
o  5937
o  5938    PROCEDURE [INLINE] mmp$touch_all_pages
o  5939      (     pages_to_touch: ^cell;
o  5940            length: ost$segment_length);
o  5941
o  5942      VAR
o  5943        temp_pages_to_touch: ^array [1 .. 7fffffff(16)] of 0 .. 0ff(16),
o  5944        page_size: ost$page_size,
o  5945        referenced_byte: 0 .. 0ff(16),
o  5946        size_offset: integer;
o  5947
o  5948      page_size := 512 * (128 - #READ_REGISTER (osc$pr_page_size_mask));
o  5949      temp_pages_to_touch := pages_to_touch;
o  5950      size_offset := 1;
o  5951
o  5952      WHILE size_offset < length DO
o  5953        referenced_byte := temp_pages_to_touch^ [size_offset];
o  5954        size_offset := size_offset + page_size;
o  5955      WHILEND;
o  5956
o  5957      referenced_byte := temp_pages_to_touch^ [length];
o  5958
o  5959   PROCEND mmp$touch_all_pages;
o  5960
o  5961
o  5962    PROCEDURE [XREF] mmp$unlock_segment (p: ^cell;
o  5963          page_disposition: mmt$lus_page_disposition;
o  5964          wait: ost$wait;
o  5965       VAR status: ost$status);
o  5966
   5969
   5970    PROCEDURE [XREF] mmp$write_modified_pages (pva: ^cell;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
          5971           length: ost$byte_count;
          5972           waitopt: ost$wait;
          5973       VAR status: ost$status);
          5974
      o   5977
      o   5978     PROCEDURE [XREF] osp$begin_system_activity;
      o   5979
      o   5980     PROCEDURE [XREF] osp$end_system_activity;
      o   5981
      o   5982     PROCEDURE [XREF] osp$set_status_abnormal ALIAS 'ospssa'
      o   5983       (     identifier: ost$status_identifier;
      o   5984             condition: ost$status_condition_code;
      o   5985             text: string ( * <= osc$max_string_size);
      o   5986        VAR status: ost$status);
      o   5987
          5994
          5995     PROCEDURE [XREF] pmp$cycle
          5996       (VAR status: ost$status);
          5997
      o   6000
      o   6001
      o   6002     PROCEDURE [XREF] pmp$delay
      o   6003       (     milliseconds: integer;
      o   6004        VAR status: ost$status);
      o   6005
          6008
          6009
          6010     PROCEDURE [INLINE] pmp$find_executing_task_xcb (VAR xcb:
          6011       ^ost$execution_control_block);
      o   6027
      o   6028     PROCEDURE [XREF] pmp$find_task_xcb (task_id: pmt$task_id;
      o   6029       VAR xcb: ^ost$execution_control_block);
          6032
          6033     PROCEDURE [INLINE] pmp$get_executing_task_gtid (VAR global_task_id:
          6034       ost$global_task_id);
      o   6053
      o   6054     PROCEDURE [XREF] pmp$set_system_flag (flag_id: ost$system_flag;
      o   6055       recipient: ost$global_task_id;
      o   6056       VAR status: ost$status);
      o   6057
          6101     PROCEDURE [XREF] pmp$zero_out_table
          6102       (     p: ^cell;
          6103             len: ost$byte_count);
          6104
      o   6107
      o   6108   PROCEDURE [XREF] syp$cause_condition (name: ost$name);
          6111
          6112     PROCEDURE [XREF] syp$mfh_for_hang_task;
          6113     PROCEDURE [INLINE] syp$set_status_from_mtr_status (monitor_status:
          6114       syt$monitor_status;
          6115       VAR status: ost$status);
          6116
      o   6125
      o   6126     PROCEDURE [XREF] syp$return_jobs_r1_resources;
      o   6127
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Procedures referenced in this module

```
      o   6128       PROCEDURE [XREF] syp$terminate_task
      o   6129         (     terminate_reason: ost$ring1_termination_reason);
      o   6130
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Variables referenced in this module

```
          6134 VAR
          6135   dmv$idle_system: [XREF] boolean;
          6136
          6137   VAR
          6138     gfv$null_sfid: [XREF, READ, OSS$MAINFRAME_WIRED_LITERAL] gft$system_file_identifier;
          6139
          6140
          6141   SECTION
          6142     oss$mainframe_wired_literal: READ;
          6143
          6144   VAR
          6145     jmv$executing_within_system_job: [XREF, oss$job_fixed] boolean;
          6146
      o   6152 {Job Control Block (JCB).}
      o   6153
      o   6154   VAR
      o   6155     jmv$jcb: [XREF] jmt$job_control_block;
          6965   VAR
          6966     jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
          6967
      o   6970   VAR
      o   6971     jmv$task_private_templ_p: [XREF] ^pmt$task_template;
          6986 {Pointer to the Active Segment Table - (AST).}
          6987
          6988   VAR
          6989     mmv$ast_p: [XREF] ^mmt$active_segment_table;
          6990
      o   6993 {This deck contains XREFs to variables used by Memory Manager in
      o   6994 {monitor mode. The variables are initialized by a job mode routine
      o   6995 {during system deadstart.
      o   6996
      o   6997   VAR
      o   6998     mmv$a_divisor: [XREF] 0 .. 10000(16),
      o   6999     mmv$a_mult: [XREF] 0 .. 10000(16),
      o   7000     mmv$number_free_astes: [XREF] integer;
      o   7001
      o   7002
      o   7003
      o   7004 {  Define variable that contains the SDT index for the first transient segment.
      o   7005
      o   7006   VAR
      o   7007     mmv$first_transient_seg_index: [XREF] ost$segment;
      o   7008
          7011 {Maximum number of pages that will be assigned to a segment that does not
          7012 {have a backing file. A signal is sent to the task to assign a file when
          7013 {the number of assigned pages exceeds this value.
          7014
          7015   VAR
          7016     mmv$max_pages_no_file: [XREF] integer;
          7017 VAR
          7018   mmv$page_map_offsets: [XREF] mmt$page_map_offsets;
          7019
      o   7043
      o   7044 { Define the number of page faults that trigger prestreaming mode
      o   7045
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
External Variables referenced in this module

```
      o   7046   VAR
      o   7047     mmv$page_streaming_prestream: [XREF] 0 .. 255;
      o   7048
      o   7049 { Define the number of bytes used to override the transfer size
      o   7050
      o   7051   VAR
      o   7052     mmv$page_streaming_transfer: [XREF] integer;
      o   7053
      o   7054   VAR
      o   7055     mmv$preset_conversion_table: [XREF, READ] array [pmt$initialization_value]
      o   7056         of integer;
      o   7057
      o   7058
      o   7059   VAR
      o   7060     mmv$shadow_by_segnum: [XREF] boolean;
      o   7061
      o   7062   VAR
      o   7063     osv$job_fixed_heap: [XREF, READ, oss$job_fixed] ^ost$heap;
      o   7064
          7067
          7068   VAR
          7069     osv$cpus_logically_on: [XREF] 0 .. osc$max_number_of_processors;
          7070
      o   7080
      o   7081   VAR
      o   7082     osv$multiple_cpus_possible: [XREF] boolean;
      o   7083
      o   7084 {System page size.}
      o   7085
      o   7086   VAR
      o   7087     osv$page_size: [XREF] ost$page_size;
      o   7088
      o   7091   VAR
      o   7092     syv$job_initialization_complete: [XREF] boolean;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
Global Declarations defined and used in this module

```
o  7094
o  7095    CONST
o  7096      max_specified_transfer_size = 1048576; { Limit user specified transfer size to 1MB (1,048,576)
o  7097
o  7098 {  Define global variables used by this module.
o  7099
o  7100    VAR
o  7101      mmv$sfid_match: [XDCL] integer,
o  7102      mmv$sfid_mismatch: [XDCL] integer,
o  7103      mmv$sparse_threshold: [XDCL] integer := 30 * 4096,  {Arbitrary number}
o  7104
o  7105      mmv$default_sdt_entry: [READ, oss$mainframe_paged_literal] mmt$segment_descriptor :=
o  7106            [[osc$v1_regular_segment, osc$non_executable, osc$read_uncontrolled, osc$write_uncontrolled, 1, 1,
o  7107            0, [FALSE, FALSE, 0]], 0, 0],
o  7108      mmv$default_sdtx_entry: [XDCL, #GATE, READ, oss$mainframe_paged_literal]
o  7109            mmt$segment_descriptor_extended := [1, mmc$sas_allow_access, * , mmc$si_none,
o  7110            mmc$srs_not_reserved, []], mmc$sar_write_extend, mmc$lss_none, [0, 0, *, mmc$ssk_none, FALSE],
o  7111            sfc$no_limit, [0, 0, 2, 0, FALSE, FALSE, FALSE], osc$max_segment_length];
o  7112
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
ASID FUNCTIONS - (from common decks)

```
o  7114
o  7115 {-------------------------------------------------------------------------------------------------------------
o  7116 {Name:
o  7117 {    mmp$ast_index
o  7118 {Purpose:
o  7119 {These functions convert AST indexes into an ASID and vise-versa.
o  7120 {Input:
o  7121 {    AST_index or ASID
o  7122 {Output:
o  7123 {    asid or ast_index
o  7124 {-------------------------------------------------------------------------------------------------------------
o  7125
o  7126   VAR
o  7127     bits: array [0 .. 15] of 0 .. 255 := [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15];
o  7128
o  7129   PROCEDURE [INLINE] mmp$asti (xasid: ost$asid;
o  7130     VAR xasti: mmt$ast_index);
o  7131
o  7132     VAR
o  7133       asid: ost$asid,
o  7134       i: integer,
o  7135       asti: integer;
o  7136
o  7137
o  7138     asid := xasid DIV mmv$a_mult + (xasid MOD mmv$a_mult) * mmv$a_divisor;
o  7139     asti := 0;
o  7140     FOR i := 1 TO 4 DO
o  7141       asti := asti * 16 + bits [asid MOD 16];
o  7142       asid := asid DIV 16;
o  7143     FOREND;
o  7144     xasti := asti;
o  7145   PROCEND mmp$asti;
o  7146
o  7147
o  7148   PROCEDURE [INLINE] mmp$asid (xasti: mmt$ast_index;
o  7149     VAR asid: ost$asid);
o  7150
o  7151     VAR
o  7152       asti: mmt$ast_index;
o  7153
o  7154     asti := xasti;
o  7155     asid := (bits [asti MOD 16] * 4096) + (bits [(asti DIV 16) MOD 16] * 256) + (bits [(asti DIV 256) MOD 16]
o  7156           * 16) + bits [(asti DIV 4096) MOD 16];
o  7157     asid := asid DIV mmv$a_divisor + (asid MOD mmv$a_divisor) * mmv$a_mult;
o  7158
o  7159   PROCEND mmp$asid;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
DESTROY_SEGMENT

```
0  7161   PROCEDURE destroy_segment
4  7162     (    xsfid: gft$system_file_identifier;
4  7163          fde_entry_p: gft$file_desc_entry_p;
4  7164          file_limits_enforced: sft$file_space_limit_kind;
4  7165      VAR status: ost$status);
4  7166
4  7167     VAR
4  7168       rb: mmt$rb_ring1_segment_request,
4  7169       sfid: gft$system_file_identifier;
4  7170
4  7171     sfid := xsfid;
4  7172     IF fde_entry_p^.media = gfc$fm_transient_segment THEN
10 7173       IF fde_entry_p^.asti <> 0 THEN
18 7174         rb.reqcode := syc$rc_ring1_segment_request;
18 7175         rb.request := mmc$sr1_delete_seg_sfid;
18 7176         rb.sfid := sfid;
18 7177         i#call_monitor (#LOC (rb), #SIZE (rb));
4A 7178       IFEND;
4A 7179       gfp$free_fde (fde_entry_p);
5C 7180     ELSE
5C 7181       dmp$destroy_file (sfid, file_limits_enforced, status);
78 7182     IFEND;
78 7183
78 7184   PROCEND destroy_segment;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
EXPAND_SEGMENT_TABLE

```
0  7186
0  7187   PROCEDURE expand_segment_table
0  7188     (    xcb_p: ^ost$execution_control_block;
0  7189      VAR status: ost$status);
0  7190
0  7191 {
0  7192 {   The purpose of this procedure is to expand the SDT and SDTX when a segment table full
0  7193 { situation is encountered when adding a new segment.  Currently, the maximum number of
0  7194 { segments a task can have open/attached is 4095.
0  7195 {
0  7196
0  7197     CONST
0  7198       segment_table_size_increase = 32;
0  7199
0  7200     VAR
0  7201       new_sdt_length: integer, {must be an integer variable}
0  7202       new_table_size: ost$segment_length,
0  7203       old_sdt_length: ost$segment,
0  7204       old_sdt_offset: 0 .. 0ffffffff(16),
0  7205       old_sdtx_offset: 0 .. 0ffffffff(16),
0  7206       old_sdt_p: ^cell,
0  7207       old_sdtx_p: ^cell,
0  7208       new_sdt_p: ^mmt$segment_descriptor_table,
0  7209       new_sdtx_p: ^mmt$segment_descriptor_table_ex,
0  7210       request_block: mmt$rb_change_segment_table;
0  7211
0  7212     status.normal := TRUE;
4  7213
4  7214 { Save the following values, so that the old SDT and SDTX can be freed after
4  7215 { the new ones are successsfully allocated.
4  7216
4  7217     old_sdt_offset := xcb_p^.sdt_offset;
4  7218     old_sdtx_offset := xcb_p^.sdtx_offset;
4  7219     old_sdt_length := xcb_p^.xp.segment_table_length;
4  7220
4  7221     new_sdt_length := xcb_p^.xp.segment_table_length + segment_table_size_increase;
4  7222
4  7223     IF ((new_sdt_length + 1) * 8) > osv$page_size THEN
32 7224       new_sdt_length := ((((new_sdt_length + 1) * 8) + osv$page_size) DIV osv$page_size);
32 7225       new_table_size := new_sdt_length * osv$page_size;
32 7226       new_sdt_length := ((new_sdt_length * osv$page_size) DIV 8) - 1;
4C 7227     IFEND;
4C 7228
4C 7229     IF new_sdt_length >= 4096 THEN
54 7230       new_sdt_length := 4095;
58 7231     IFEND;
58 7232
58 7233     IF new_sdt_length = xcb_p^.xp.segment_table_length THEN
64 7234       osp$set_status_abnormal ('MM', mme$segment_table_is_full, '', status);
96 7235       RETURN;
98 7236     IFEND;
98 7237
98 7238     ALLOCATE new_sdt_p: [0 .. new_sdt_length] IN osv$job_fixed_heap^;
D4 7239
D4 7240     IF ((new_sdt_length + 1) * 8) > osv$page_size THEN
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
EXPAND_SEGMENT_TABLE

```
 DC  7241        mmp$free_pages (new_sdt_p, new_table_size, osc$nowait, status);
100  7242        mmp$assign_contiguous_memory (new_sdt_p, new_table_size, status);
11C  7243        IF NOT status.normal THEN
124  7244          RETURN;
126  7245        IFEND;
12A  7246      ELSE
12A  7247        pmp$zero_out_table (#LOC (new_sdt_p^), #SIZE (new_sdt_p^));
146  7248      IFEND;
146  7249
146  7250 {  Allocate and zero out the SDTX.
146  7251
146  7252      ALLOCATE new_sdtx_p: [0 .. new_sdt_length] IN osv$job_fixed_heap^;
186  7253
186  7254      pmp$zero_out_table (#LOC (new_sdtx_p^), #SIZE (new_sdtx_p^));
1A2  7255
1A2  7256 {  Issue monitor request to move old segment table to new segment table, update segment table
1A2  7257 {  address and segment table length in task's exchange package.
1A2  7258
1A2  7259      request_block.request_code := syc$rc_change_segment_table;
1A2  7260      request_block.new_sdt_offset := #OFFSET (new_sdt_p);
1A2  7261      request_block.new_sdtx_offset := #OFFSET (new_sdtx_p);
1A2  7262      request_block.new_sdt_length := new_sdt_length;
1A2  7263      i#call_monitor (#LOC (request_block), #SIZE (request_block));
1E0  7264      syp$set_status_from_mtr_status (request_block.status, status);
220  7265      IF NOT status.normal THEN
228  7266        osp$system_error ('Error in change segment table monitor request.', ^status);
24C  7267        RETURN;
24E  7268      IFEND;
24E  7269
24E  7270 {  Free the old SDT and SDTX tables.
24E  7271 {  NOTE: Job monitor's SDT and SDTX are not allocated, hence they can not be freed unless they
24E  7272 {        have been expanded once.
24E  7273
24E  7274      IF (old_sdt_length * 8) > osv$page_size THEN
25A  7275        mmp$free_pages (#ADDRESS (1, osc$segnum_job_fixed_heap, old_sdtx_offset),
290  7276              (old_sdt_length * #SIZE (mmt$segment_descriptor_extended)), osc$wait, status);
290  7277      IFEND;
290  7278      IF old_sdt_offset >= #OFFSET (osv$job_fixed_heap) THEN
29C  7279        old_sdt_p := #ADDRESS (1, osc$segnum_job_fixed_heap, old_sdt_offset);
29C  7280        old_sdtx_p := #ADDRESS (1, osc$segnum_job_fixed_heap, old_sdtx_offset);
29C  7281        FREE old_sdt_p IN osv$job_fixed_heap^;
2D4  7282        FREE old_sdtx_p IN osv$job_fixed_heap^;
2F6  7283      IFEND;
2F6  7284
2F6  7285    PROCEND expand_segment_table;
  0  7286
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
FIND_AVAILABLE_SEGMENT_NUMBER

```
  0  7288
  0  7289    PROCEDURE find_available_segment_number
  0  7290      (     xcb_p: ^ost$execution_control_block;
  0  7291            segment_res_state: mmt$segment_reservation_state;
  0  7292       VAR segnum: ost$segment;
  0  7293       VAR status: ost$status);
  0  7294
  0  7295      VAR
  0  7296        sdt_p: mmt$max_sdt_p,
  0  7297        sdtx_p: mmt$max_sdtx_p,
  0  7298        segment_table_length: integer;
  0  7299
  0  7300
  0  7301      {  Find an available segment number.
  0  7302
  0  7303      status.normal := TRUE;
  4  7304      segnum := mmv$first_transient_seg_index - 1;
  4  7305      segment_table_length := xcb_p^.xp.segment_table_length;
  4  7306      mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
  4  7307
  4  7308      REPEAT
 54  7309        segnum := segnum + 1;
 54  7310        IF segnum > segment_table_length THEN
 62  7311          expand_segment_table (xcb_p, status);
 76  7312          IF NOT status.normal THEN
 7E  7313            RETURN;
 80  7314          IFEND;
 80  7315          segment_table_length := xcb_p^.xp.segment_table_length;
 8E  7316        IFEND;
 8E  7317      UNTIL (sdt_p^.st [segnum].ste.vl = osc$vl_invalid_entry) AND
 C0  7318            (sdtx_p^.sdtx_table [segnum].segment_reservation_state = segment_res_state);
 C0  7319
 C0  7320    PROCEND find_available_segment_number;
  0  7321
```

```
      0  7323
      0  7324   PROCEDURE store_ste_in_segment_table
      0  7325     (    xsdt_entry: mmt$segment_descriptor;
      0  7326          sfid: gft$system_file_identifier;
      0  7327          ste_p: ^mmt$segment_descriptor;
      0  7328          fde_entry_p: gft$locked_file_desc_entry_p;
      0  7329          segnum: ost$segment);
      0  7330
      0  7331 { This routine is tricky so make sure you understand it before changing it!!!!!!!!!!!
      0  7332 { This routine stores a STE entry into the segment table. If the ASID in the new ste
      0  7333 { entry is zero then the procedure is straightforward.
      0  7334 { If, however, the STE already has an ASID then things get more complicated since monitor may be
      0  7335 { changing the ASID asynchronously while this routine is running.
      0  7336 {        . NEVER let a STE entry get into the segment table if the ASID/ASTI dont match
      0  7337 {        . after putting an entry in with a non-zero ASID, check the AST to see if the AST.SFID
      0  7338 {          agrees with the SDTX.SFID. IF they agree all is well. Otherwise, clear the ASID/ASTI to zero;
      0  7339 {          the assumption is that Memory Manager changed the ASID. The correct ASID will be fetched
      0  7340 {          on the first page fault. (this should not happen very much.)
      0  7341
      0  7342     VAR
      0  7343       aste_p: ^mmt$active_segment_table_entry,
      0  7344       asid: ost$asid,
      0  7345       asti: mmt$ast_index,
      0  7346       rb: mmt$rb_ring1_segment_request,
      0  7347       sdt_entry: mmt$segment_descriptor;
      0  7348
      0  7349     sdt_entry := xsdt_entry;
      4  7350     asti := fde_entry_p^.asti;
      4  7351     IF (asti = 0) OR (sdt_entry.ste.asid = OFFFF(16)) OR (mmv$ast_p = NIL) THEN
     2C  7352       ste_p^ := sdt_entry;
     32  7353     ELSE
     32  7354       sdt_entry.asti := asti;
     32  7355       mmp$asid (asti, asid);
     32  7356       sdt_entry.ste.asid := asid;
     32  7357       ste_p^ := sdt_entry;
     32  7358       aste_p := ^mmv$ast_p^ [sdt_entry.asti];
     32  7359
     32  7360       IF (aste_p^.in_use) AND (aste_p^.sfid = sfid) AND
    172  7361            (((sfid.residence = gfc$tr_job) AND (aste_p^.ijl_ordinal = jmv$jcb.ijl_ordinal)) OR
    172  7362            ((sfid.residence = gfc$tr_system) AND
    172  7363            (((aste_p^.queue_id = mmc$pq_job_working_set) AND (aste_p^.ijl_ordinal = jmv$jcb.ijl_ordinal) AND
    172  7364            ((fde_entry_p^.queue_status = gfc$qs_job_working_set) OR
    172  7365            ((fde_entry_p^.queue_status = gfc$qs_job_shared) AND (fde_entry_p^.attach_count = 1)))) OR
    172  7366            (((aste_p^.queue_id >= mmc$pq_shared_first) AND (aste_p^.queue_id <= mmc$pq_shared_last)) AND
    172  7367            (aste_p^.ijl_ordinal = jmv$system_ijl_ordinal) AND
    172  7368            ((fde_entry_p^.queue_status = gfc$qs_global_shared) OR
    172  7369            ((fde_entry_p^.queue_status = gfc$qs_job_shared) AND (fde_entry_p^.attach_count > 1)))))))) THEN
    172  7370
    172  7371         mmv$sfid_match := mmv$sfid_match + 1;
    17E  7372       ELSE
    17E  7373         mmv$sfid_mismatch := mmv$sfid_mismatch + 1;
    17E  7374         ste_p^ := xsdt_entry;
    17E  7375
    17E  7376 { If the file is job shared and there is more that one user (attach) of the file, the above tests which
    17E  7377 { would have allowed the asid to be stored may have failed because pages of the file are being kept in
```



```
    17E  7378 { the working set and should now be kept in the shared queue.  (Or they may have failed because the
    17E  7379 { asid changed--we can't be sure by looking at the ast.queue_id.)  Issue a monitor request to straighten
    17E  7380 { out job shared files.  If there is more than one user, pages of job shared files must be removed
    17E  7381 { from the jws of the original job and moved to the shared queue before a second user can reference
    17E  7382 { any of the pages.
    17E  7383
    17E  7384         IF (fde_entry_p^.queue_status = gfc$qs_job_shared) AND (fde_entry_p^.attach_count > 1) THEN
    19E  7385           rb.reqcode := syc$rc_ring1_segment_request;
    19E  7386           rb.request := mmc$sr1_remove_job_shared_pages;
    19E  7387           rb.system_file_id := sfid;
    19E  7388           rb.segment_number := segnum;
    19E  7389           rb.server_file := FALSE;
    19E  7390           i#call_monitor (#LOC (rb), #SIZE (rb));
    1D6  7391         IFEND;
    1D6  7392       IFEND;
    1D6  7393     IFEND;
    1D6  7394   PROCEND store_ste_in_segment_table;
      0  7395
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
ADD_SDT_SDTX_ENTRY

```
0  7397
0  7398 {
0  7399 {   The purpose of this procedure is to add the SDT and SDTX entries to the
0  7400 { SDT and the SDTX tables respectively.  The caller can specify a segment
0  7401 { number to use or this procedure will use the first available one.
0  7402 {
0  7403 {         MMP$ADD_SDT_SDTX_ENTRY (SDT_ENTRY, SDTX_ENTRY, FDE_ENTRY,
0  7404 {           SHARED_TASKID_ARRAY, SEGMENT_NUMBER, STATUS)
0  7405 {
0  7406 { SDT_ENTRY: (input) This parameter is the SDT entry to be added to the
0  7407 {         SDT table.
0  7408 {
0  7409 { SDTX_ENTRY: (input) This parameter is the SDTX entry to be added to the
0  7410 {         SDTX table.
0  7411 {
0  7412 { FDE_ENTRY: (input) This parameter is the FDE entry for the segment being
0  7413 {         added.
0  7414 {
0  7415 { SHARED_TASKID_ARRAY: (input) This parameter specifies the list of task-ids
0  7416 {         in which this segment should be added. Currently, the only user of
0  7417 {         this capability is ADA-TASKING. All other callers should pass a NIL
0  7418 {         pointer for this parameter.
0  7419 {
0  7420 { SEGMENT_NUMBER: (input) This parameter specifies the segment number which
0  7421 {         the user passed or the system assigned for this instance of open.
0  7422 {
0  7423 { STATUS: (output) This parameter is where the request status is returned
0  7424 {         to the caller.
0  7425 {               dme$unable_to_locate_fde
0  7426 {               dme$unable_to_get_fd_lock
0  7427 {               mme$contig_mem_seg_violation
0  7428 {               mme$invalid_length_requested
0  7429 {               mme$invalid_pva
0  7430 {               mme$invalid_ring_brackets
0  7431 {               mme$invalid_shared_taskid
0  7432 {               mme$pages_already_assigned
0  7433 {               mme$ref_to_unrecovered_file
0  7434 {               mme$ring_violation
0  7435 {               mme$segment_number_is_in_use
0  7436 {               mme$segment_number_not_in_use
0  7437 {               mme$segment_number_too_big
0  7438 {               mme$segment_origin_invalid
0  7439 {               mme$segment_table_is_full
0  7440 {               mme$unable_to_assign_contig_mem
0  7441 {
0  7442
0  7443
0  7444
0  7445
0  7446
0  7447   PROCEDURE add_sdt_sdtx_entry
0  7448     (     sdt_entry: mmt$segment_descriptor;
0  7449           sdtx_entry: mmt$segment_descriptor_extended;
0  7450           fde_entry_p: gft$locked_file_desc_entry_p;
0  7451           shared_taskid_array: ^array [1 .. * ] of pmt$task_id;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
ADD_SDT_SDTX_ENTRY

```
0  7452           segment_number: ost$segment);
0  7453
0  7454      VAR
0  7455        cache_bypass: boolean,
0  7456        cell_p: ^cell,
0  7457        i: integer,
0  7458        local_sdt: mmt$segment_descriptor,
0  7459        local_sdtx: mmt$segment_descriptor_extended,
0  7460        pva_p: ^ost$pva,
0  7461        sdt_entry_p: ^mmt$segment_descriptor,
0  7462        sdtx_entry_p: ^mmt$segment_descriptor_extended,
0  7463        shadow_fde_p: gft$locked_file_desc_entry_p,
0  7464        task_sdt_p: ^mmt$segment_descriptor,
0  7465        task_sdtx_p: ^mmt$segment_descriptor_extended,
0  7466        task_xcb: ^ost$execution_control_block,
0  7467        xcb_p: ^ost$execution_control_block;
0  7468
0  7469      local_sdt := sdt_entry;
4  7470      local_sdtx := sdtx_entry;
E  7471
E  7472      pmp$find_executing_task_xcb (xcb_p);
14 7473
14 7474 { Set cache bypass if required for multiprocessing.
14 7475
14 7476      cache_bypass := FALSE;
14 7477      IF osv$multiple_cpus_possible THEN
3A 7478        IF (local_sdt.ste.xp = osc$non_executable) OR ((local_sdt.ste.xp <> osc$non_executable) AND
4A 7479              (local_sdt.ste.wp <> osc$non_writable)) THEN
4A 7480
4A 7481 { I dont think we need a clause for global_unnamed files because it appears
4A 7482 { that they would drop out anyway.
4A 7483
4A 7484          IF (fde_entry_p^.file_kind <> gfc$fk_unnamed_file) THEN
54 7485            IF (fde_entry_p^.queue_status = gfc$qs_job_working_set) THEN
5E 7486              IF (jmv$jcb.ijle_p^.multiprocessing_allowed) THEN
6E 7487                cache_bypass := TRUE;
6E 7488                local_sdt.ste.v1 := osc$v1_cache_bypass;
78 7489              IFEND;
7C 7490            ELSE
7C 7491              IF (fde_entry_p^.queue_status = gfc$qs_global_shared) THEN
80 7492                cache_bypass := TRUE;
80 7493                local_sdt.ste.v1 := osc$v1_cache_bypass;
8A 7494              IFEND;
8A 7495            IFEND;
8A 7496          IFEND;
8A 7497        IFEND;
8A 7498      IFEND;
8A 7499
8A 7500
8A 7501 {  Add sdt_entry to the task's segment descriptor table (SDT) and the sdtx_entry to the
8A 7502 {  segment descriptor table extended (SDTX).
8A 7503
8A 7504      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segment_number);
8A 7505      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segment_number);
8A 7506      local_sdtx.segment_reservation_state := sdtx_entry_p^.segment_reservation_state;
```

```
      8A  7507      mmp$set_segment_access_rights (local_sdt, local_sdtx);
     10A  7508
     10A  7509      sdtx_entry_p^ := local_sdtx;
*WARN*    7510      store_ste_in_segment_table (local_sdt, local_sdtx.sfid, sdt_entry_p, fde_entry_p, segment_number);
     13A  7511      fde_entry_p^.open_count := fde_entry_p^.open_count + 1;
     13A  7512
     13A  7513      IF local_sdtx.shadow_info.shadow_segment_kind <> mmc$ssk_none THEN
     14E  7514        gfp$get_locked_fde_p (local_sdtx.shadow_info.shadow_sfid, shadow_fde_p);
     284  7515        shadow_fde_p^.open_count := shadow_fde_p^.open_count + 1;
     284  7516        gfp$unlock_fde_p (shadow_fde_p);
     384  7517      IFEND;
     384  7518
     384  7519 {If a stack segment was just created, update the TOS pointer.
     384  7520
     384  7521      IF mmc$sa_stack IN sdtx_entry.software_attribute_set THEN
     392  7522        cell_p := #ADDRESS (local_sdt.ste.r1, segment_number,
     392  7523             (mmv$page_map_offsets [mmc$pmo_user_stack] * osv$page_size) + mmc$ring_crossing_offset);
     392  7524        pva_p := #LOC (cell_p);
     3CA  7525        xcb_p^.xp.tos_registers [local_sdt.ste.r1].pva := pva_p^;
     3CA  7526        fde_entry_p^.stack_for_ring := local_sdt.ste.r1;
     3E6  7527      IFEND;
     3E6  7528
     3E6  7529      IF shared_taskid_array <> NIL THEN
     3F4  7530
     3F4  7531        { Mmc$sa_stack is removed from the software attribute set before the sdtx entry of the segment
     3F4  7532        { is propagated to all of the other tasks. The task which opens the stack segment will be
     3F4  7533        { the only task with mmc$sa_stack in its sdtx entry.
     3F4  7534
     3F4  7535        local_sdtx.software_attribute_set := local_sdtx.software_attribute_set -
     3F4  7536             $mmt$software_attribute_set [mmc$sa_stack];
     3F4  7537        FOR i := LOWERBOUND (shared_taskid_array^) TO UPPERBOUND (shared_taskid_array^) DO
     410  7538          pmp$find_task_xcb (shared_taskid_array^ [i], task_xcb);
     436  7539          IF task_xcb <> xcb_p THEN
     446  7540            task_sdt_p := mmp$get_sdt_entry_p (task_xcb, segment_number);
     446  7541            task_sdtx_p := mmp$get_sdtx_entry_p (task_xcb, segment_number);
     446  7542            task_sdtx_p^ := local_sdtx;
     4A2  7543            mmp$set_segment_access_rights (local_sdt, task_sdtx_p^);
*WARN*    7544            store_ste_in_segment_table (local_sdt, local_sdtx.sfid, task_sdt_p, fde_entry_p, segment_number);
     508  7545            fde_entry_p^.open_count := fde_entry_p^.open_count + 1;
     51A  7546          IFEND;
     51A  7547        FOREND;
     520  7548      IFEND;
     520  7549
     520  7550      IF (osv$cpus_logically_on > 1) AND (NOT cache_bypass) THEN
     530  7551        cell_p := #ADDRESS (1, segment_number, 0);
     530  7552        #PURGE_BUFFER (osc$pva_purge_segment_cache, cell_p);
     548  7553      IFEND;
     548  7554
     548  7555    PROCEND add_sdt_sdtx_entry;
       0  7556
```



```
       0  7558 {
       0  7559 { Purpose:
       0  7560 {    This procedure assigns disk space for all pages currently assigned to a segment/file.
       0  7561 {    If necessary, it converts a transient file into a disk file.
       0  7562 {
       0  7563 {  segment_number: If non zero, this specifies the segment number
       0  7564 {  sfid: If segment number is zero, this is the SFID of the file
       0  7565 {  min_allocation_length: Normally zero. If zero disk space is assigned to all
       0  7566 {     pages that exist in the file. If non-zero, contiguous disk space is assign for
       0  7567 {     offset zero thru this offset. NOTE if
       0  7568 {
       0  7569 {
       0  7570
       0  7571
       0  7572   PROCEDURE [XDCL, #GATE] mmp$assign_mass_storage
       0  7573     (     segment_number: ost$segment;
       0  7574           xsfid: gft$system_file_identifier;
       0  7575           min_allocation_length: ost$segment_length;
       0  7576      VAR status: ost$status);
       0  7577
       0  7578     VAR
       0  7579       assign_active: amt$file_byte_address,
       0  7580       fde_p: gft$locked_file_desc_entry_p,
       0  7581       file_attributes_p: ^array [1 .. * ] of dmt$file_attribute,
       0  7582       file_limits: sft$file_space_limit_kind,
       0  7583       length_to_allocate: ost$segment_length,
       0  7584       max_pages_no_file: integer,
       0  7585       page_streaming_ts_shift: 0 .. 15,
       0  7586       sdt_entry_p: ^mmt$segment_descriptor,
       0  7587       sdtx_entry_p: ^mmt$segment_descriptor_extended,
       0  7588       sfid: gft$system_file_identifier,
       0  7589       xcb_p: ^ost$execution_control_block;
       0  7590
       0  7591
       0  7592     status.normal := TRUE;
       A  7593
       A  7594     pmp$find_executing_task_xcb (xcb_p);
      1A  7595     IF segment_number <> 0 THEN
      36  7596       sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segment_number);
      36  7597       file_limits := sdtx_entry_p^.file_limits_enforced;
      36  7598       assign_active := sdtx_entry_p^.assign_active;
      36  7599       sfid := sdtx_entry_p^.sfid;
      36  7600       IF (mmc$sa_wired IN sdtx_entry_p^.software_attribute_set) OR
      7C  7601            (mmc$sa_fixed IN sdtx_entry_p^.software_attribute_set) THEN
      7C  7602         osp$set_status_abnormal ('MM', mme$segment_not_pageable, '', status);
      AE  7603         RETURN;
      B0  7604       IFEND;
      B4  7605     ELSE
      B4  7606       file_limits := sfc$no_limit;
      B4  7607       assign_active := mmc$assign_active_escaped;
      B4  7608       sfid := xsfid;
      C8  7609     IFEND;
      C8  7610
      C8  7611
      C8  7612     gfp$get_locked_fde_p (sfid, fde_p);
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$ASSIGN_MASS_STORAGE

```
1F2  7613
1F2  7614      IF NOT dmv$idle_system THEN
1FE  7615        IF fde_p^.media = gfc$fm_transient_segment THEN
20A  7616          PUSH file_attributes_p: [1 .. 1];
224  7617          file_attributes_p^ [1].keyword := dmc$class;
224  7618          IF jmv$executing_within_system_job THEN
23A  7619            file_attributes_p^ [1].class := rmc$msc_system_critical_files;
246  7620          ELSE
246  7621            file_attributes_p^ [1].class := rmc$msc_user_temporary_files;
24E  7622          IFEND;
24E  7623          dmp$create_disk_file (fde_p, file_attributes_p, 0, sfid, status);
27C  7624          IF (assign_active <> mmc$assign_active_escaped) AND
292  7625            NOT (sdtx_entry_p^.stream.transfer_size_specified) THEN
292  7626            mmp$convert_ps_transfer_size (fde_p^.transfer_unit_size, page_streaming_ts_shift);
2D2  7627            sdtx_entry_p^.stream.transfer_size := page_streaming_ts_shift;
2E2  7628          IFEND;
2E2  7629        IFEND;
2E2  7630
2E2  7631        IF status.normal THEN
2EE  7632          length_to_allocate := fde_p^.eoi_byte_address;
2EE  7633          IF (min_allocation_length > 0) OR (length_to_allocate < mmv$sparse_threshold) OR
314  7634              (fde_p^.media = gfc$fm_served_file) THEN
314  7635            IF length_to_allocate < min_allocation_length THEN
31C  7636              length_to_allocate := min_allocation_length;
320  7637            IFEND;
320  7638            dmp$allocate_file_space_r1 (sfid, 0, length_to_allocate, 0, osc$nowait, file_limits, status);
34E  7639          ELSE
34E  7640            dmp$sparse_allocate (sfid, assign_active, file_limits, status);
36C  7641          IFEND;
36C  7642        IFEND;
36C  7643      IFEND;
36C  7644
36C  7645      gfp$unlock_fde_p (fde_p);
460  7646
460  7647  { If everything worked OK, clear the assign active flag in the SDTX or XCB since there cannot
460  7648  { be any more escaped allocation or allocation required.
460  7649
460  7650      IF status.normal THEN
46C  7651        IF segment_number <> 0 THEN
474  7652          sdtx_entry_p^.assign_active := mmc$assign_active_null;
47E  7653        ELSE
47E  7654          xcb_p^.assign_active_sfid := gfv$null_sfid;
48E  7655        IFEND;
48E  7656      IFEND
48E  7657
48E  7658    PROCEND mmp$assign_mass_storage;
  0  7659
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$BUILD_SEGMENT

```
  0  7661
  0  7662    PROCEDURE [XDCL, #GATE] mmp$build_segment
  0  7663      (     attrib: mmt$segment_attrib_descriptor;
  0  7664            shared_taskid_array: ^array [1 .. * ] of pmt$task_id;
  0  7665       VAR segment_pointer: mmt$segment_pointer;
  0  7666       VAR status: ost$status);
  0  7667
  0  7668      VAR
  0  7669        i: integer,
  0  7670        fde_entry_p: gft$file_desc_entry_p,
  0  7671        file_hash: 0 .. 255,
  0  7672        locked_fde_entry_p: gft$locked_file_desc_entry_p,
  0  7673        page_streaming_ts_shift: 0 .. 15,
  0  7674        page_streaming_transfer_size: integer,
  0  7675        sdt_entry: mmt$segment_descriptor,
  0  7676        sdtx_entry: mmt$segment_descriptor_extended,
  0  7677        sdt_entry_p: ^mmt$segment_descriptor,
  0  7678        sdtx_entry_p: ^mmt$segment_descriptor_extended,
  0  7679        segment_length: ost$segment_length,
  0  7680        segment_number: ost$segment,
  0  7681        segment_res_state: mmt$segment_reservation_state,
  0  7682        sfid: gft$system_file_identifier,
  0  7683        shadow_fde_p: gft$file_desc_entry_p,
  0  7684        shadow_sdt_p: ^mmt$segment_descriptor,
  0  7685        shadow_sdtx_p: ^mmt$segment_descriptor_extended,
  0  7686        task_xcb: ^ost$execution_control_block,
  0  7687        xcb_p: ^ost$execution_control_block;
  0  7688
  0  7689      status.normal := TRUE;
  4  7690      sdt_entry := mmv$default_sdt_entry;
  4  7691      sdtx_entry := mmv$default_sdtx_entry;
 20  7692      sdt_entry.ste.r1 := attrib.validating_ring_number;
 20  7693      sdt_entry.ste.r2 := attrib.validating_ring_number;
 20  7694      sdtx_entry.open_validating_ring_number := attrib.validating_ring_number;
 20  7695      sdtx_entry.file_limits_enforced := attrib.file_limits_to_enforce;
 20  7696      segment_number := 0;
 20  7697      IF shared_taskid_array <> NIL THEN
 54  7698        FOR i := LOWERBOUND (shared_taskid_array^) TO UPPERBOUND (shared_taskid_array^) DO
 62  7699          pmp$find_task_xcb (shared_taskid_array^ [i], task_xcb);
 88  7700          IF task_xcb = NIL THEN
 92  7701            osp$set_status_abnormal ('MM', mme$invalid_shared_taskid, '', status);
 C2  7702            RETURN;
 C4  7703          IFEND;
 C4  7704        FOREND;
 CA  7705      IFEND;
 CA  7706      IF attrib.sfid = gfv$null_sfid THEN
 DE  7707        gfp$assign_fde (gfc$tr_job, segment_number, sfid, fde_entry_p);
108  7708        IF fde_entry_p = NIL THEN
112  7709          osp$set_status_abnormal ('MM', mme$unable_to_assign_fde, '', status);
142  7710          RETURN;
144  7711        IFEND;
144  7712        gfp$get_locked_fde_p (sfid, locked_fde_entry_p);
274  7713        locked_fde_entry_p^.queue_status := gfc$qs_job_working_set;
274  7714        locked_fde_entry_p^.file_kind := gfc$fk_unnamed_file;
286  7715      ELSE
```

```
286  7716        sfid := attrib.sfid;
286  7717        gfp$get_locked_fde_p (sfid, locked_fde_entry_p);
3B8  7718        IF locked_fde_entry_p = NIL THEN
3C2  7719          osp$set_status_abnormal ('MM', mme$invalid_sfid, '', status);
3F6  7720          RETURN;
3F8  7721        IFEND;
3FA  7722      IFEND;
3FA  7723
3FA  7724      sdtx_entry.sfid := sfid;
3FA  7725
3FA  7726    /user_attributes/
3FA  7727      BEGIN
3FA  7728        IF attrib.user_attributes <> NIL THEN
410  7729          FOR i := LOWERBOUND (attrib.user_attributes^) TO UPPERBOUND (attrib.user_attributes^) DO
42A  7730            CASE attrib.user_attributes^ [i].keyword OF
4BE  7731            = mmc$kw_null_keyword =
4C2  7732            = mmc$kw_ring_numbers =
4C2  7733              IF (attrib.user_attributes^ [i].r1 > attrib.user_attributes^ [i].r2) OR
4D2  7734                   (attrib.user_attributes^ [i].r2 = 0) THEN
4D2  7735                osp$set_status_abnormal ('MM', mme$invalid_ring_brackets, '', status);
506  7736                EXIT /user_attributes/;
50A  7737              IFEND;
50A  7738              sdt_entry.ste.r1 := attrib.user_attributes^ [i].r1;
50A  7739              sdt_entry.ste.r2 := attrib.user_attributes^ [i].r2;
516  7740            = mmc$kw_segment_number =
516  7741              segment_number := attrib.user_attributes^ [i].segnum;
522  7742            = mmc$kw_hardware_attributes =
522  7743              IF mmc$ha_read IN attrib.user_attributes^ [i].hardware_attri_set THEN
540  7744                sdt_entry.ste.rp := osc$read_uncontrolled;
548  7745              ELSE
548  7746                sdt_entry.ste.rp := osc$non_readable;
54C  7747              IFEND;
54C  7748              IF mmc$ha_binding IN attrib.user_attributes^ [i].hardware_attri_set THEN
56A  7749                IF attrib.validating_ring_number <= 3 THEN
574  7750                  sdt_entry.ste.rp := osc$binding_segment;
57C  7751                ELSE
57C  7752                  osp$set_status_abnormal ('MM', mme$binding_attribute_invalid, '', status);
5B0  7753                  EXIT /user_attributes/;
5B4  7754                IFEND;
5B4  7755              IFEND;
5B4  7756              IF mmc$ha_write IN attrib.user_attributes^ [i].hardware_attri_set THEN
5D2  7757                sdt_entry.ste.wp := osc$write_uncontrolled;
5DA  7758              ELSE
5DA  7759                sdt_entry.ste.wp := osc$non_writable;
5DE  7760              IFEND;
5DE  7761              IF mmc$ha_cache_bypass IN attrib.user_attributes^ [i].hardware_attri_set THEN
5FC  7762                sdt_entry.ste.vl := osc$vl_cache_bypass;
606  7763              ELSE
606  7764                sdt_entry.ste.vl := osc$vl_regular_segment;
60A  7765              IFEND;
60A  7766              IF mmc$ha_execute IN attrib.user_attributes^ [i].hardware_attri_set THEN
628  7767                sdt_entry.ste.xp := osc$non_privileged;
630  7768              ELSE
630  7769                IF mmc$ha_execute_local IN attrib.user_attributes^ [i].hardware_attri_set THEN
638  7770                  sdt_entry.ste.xp := osc$local_privilege;
```



```
640  7771                ELSE
640  7772                  IF (mmc$ha_execute_global IN attrib.user_attributes^ [i].hardware_attri_set) AND
650  7773                       (attrib.validating_ring_number > 1) THEN
650  7774                    osp$set_status_abnormal ('MM', mme$execute_global_invalid, '', status);
684  7775                    EXIT /user_attributes/;
68C  7776                  ELSE
68C  7777                    sdt_entry.ste.xp := osc$non_executable;
690  7778                  IFEND;
690  7779                IFEND;
690  7780              IFEND;
694  7781            = mmc$kw_software_attributes =
694  7782              IF attrib.validating_ring_number <= 3 THEN
69E  7783                sdtx_entry.software_attribute_set := attrib.user_attributes^ [i].software_attri_set;
6BC  7784              ELSEIF ((attrib.validating_ring_number <= 6) AND ([$mmt$software_attribute_set [mmc$sa_stack] *
70A  7785                   attrib.user_attributes^ [i].software_attri_set) <> $mmt$software_attribute_set [])) OR
70A  7786                   ((attrib.validating_ring_number > 6) AND ([$mmt$software_attribute_set
70A  7787                   [mmc$sa_wired, mmc$sa_fixed, mmc$sa_stack] * attrib.user_attributes^ [i].
70A  7788                   software_attri_set) <> $mmt$software_attribute_set [])) THEN
70A  7789                osp$set_status_abnormal ('MM', mme$software_attribute_invalid, '', status);
73E  7790                EXIT /user_attributes/;
746  7791              ELSE
746  7792                sdtx_entry.software_attribute_set := attrib.user_attributes^ [i].software_attri_set;
760  7793              IFEND;
764  7794            = mmc$kw_error_exit_procedure =
764  7795              osp$set_status_abnormal ('MM', mme$unsupported_keyword, '', status);
798  7796              EXIT /user_attributes/;
7A0  7797            = mmc$kw_max_segment_length =
7A0  7798              locked_fde_entry_p^.file_limit := attrib.user_attributes^ [i].max_length;
7B0  7799            = mmc$kw_gl_key =
7B0  7800              sdt_entry.ste.key_lock := attrib.user_attributes^ [i].gl_key;
7CE  7801            = mmc$kw_clear_space =
7D2  7802            = mmc$kw_preset_value =
7D2  7803              locked_fde_entry_p^.preset_value := attrib.user_attributes^ [i].preset_value;
7F4  7804            = mmc$kw_segment_access_control =
7F4  7805              IF attrib.user_attributes^ [i].access_control.cache_bypass = TRUE THEN
80E  7806                sdt_entry.ste.vl := osc$vl_cache_bypass;
818  7807              ELSE
818  7808                sdt_entry.ste.vl := osc$vl_regular_segment;
81C  7809              IFEND;
81C  7810
81C  7811              IF (attrib.user_attributes^ [i].access_control.execute_privilege = osc$global_privilege) AND
840  7812                   (attrib.validating_ring_number > 1) THEN
840  7813                osp$set_status_abnormal ('MM', mme$execute_global_invalid, '', status);
874  7814                EXIT /user_attributes/;
87C  7815              ELSE
87C  7816                sdt_entry.ste.xp := attrib.user_attributes^ [i].access_control.execute_privilege;
896  7817              IFEND;
896  7818
896  7819              IF (attrib.user_attributes^ [i].access_control.read_privilege = osc$binding_segment) AND
8BA  7820                   (attrib.validating_ring_number > 3) THEN
8BA  7821                osp$set_status_abnormal ('MM', mme$binding_attribute_invalid, '', status);
8EE  7822                EXIT /user_attributes/;
8F2  7823              IFEND;
8F2  7824              sdt_entry.ste.rp := attrib.user_attributes^ [i].access_control.read_privilege;
8F2  7825              sdt_entry.ste.wp := attrib.user_attributes^ [i].access_control.write_privilege;
```

```
      918  7826            = mmc$kw_asid =
      918  7827              IF attrib.validating_ring_number <> 1 THEN
      920  7828                osp$set_status_abnormal ('MM', mme$asid_specified, '', status);
      954  7829                EXIT /user_attributes/;
      958  7830              IFEND;
      958  7831
      958  7832              IF (attrib.user_attributes^ [i].asid = osc$asid_ei) OR
      98A  7833                   (attrib.user_attributes^ [i].asid = osc$asid_eie) OR
      98A  7834                   (attrib.user_attributes^ [i].asid = osc$asid_nos) THEN
      98A  7835                sdt_entry.ste.asid := attrib.user_attributes^ [i].asid;
      9A8  7836              ELSE
      9A8  7837                osp$set_status_abnormal ('MM', mme$invalid_asid_specified, '', status);
      9D8  7838                EXIT /user_attributes/;
      9DC  7839              IFEND;
      9E0  7840            = mmc$kw_wired_segment =
      9E0  7841              sdtx_entry.software_attribute_set := sdtx_entry.software_attribute_set +
      9E0  7842                   $mmt$software_attribute_set [mmc$sa_wired];
      9E0  7843              locked_fde_entry_p^.file_limit := attrib.user_attributes^ [i].wired_segment_length;
      A10  7844            = mmc$kw_inheritance =
      A10  7845              sdtx_entry.inheritance := attrib.user_attributes^ [i].inheritance;
      A2E  7846            = mmc$kw_shadow_segment =
      A2E  7847              IF ((attrib.user_attributes^ [i].shadow_length MOD mmc$shadow_allocation_size) <> 0) THEN
      A4C  7848                osp$set_status_abnormal ('MM', mme$length_not_0_mod_16384, '', status);
      A80  7849                EXIT /user_attributes/;
      A84  7850              IFEND;
      A84  7851              IF ((#OFFSET (attrib.user_attributes^ [i].shadow_p) MOD mmc$shadow_allocation_size) <> 0) THEN
      AA0  7852                osp$set_status_abnormal ('MM', mme$address_not_0_mod_16384, '', status);
      AD4  7853                EXIT /user_attributes/;
      AD8  7854              IFEND;
      AD8  7855              sdtx_entry.shadow_info.shadow_length_page_count :=
      AD8  7856                   attrib.user_attributes^ [i].shadow_length DIV osv$page_size;
      AD8  7857              sdtx_entry.shadow_info.shadow_start_page_number := #OFFSET (attrib.user_attributes^ [i].
      AD8  7858                   shadow_p) DIV osv$page_size;
      AD8  7859              mmp$validate_segment_number (#SEGMENT (attrib.user_attributes^ [i].shadow_p),
      B1A  7860                   shadow_sdt_p, shadow_sdtx_p, status);
      B1A  7861              IF NOT status.normal THEN
      B22  7862                EXIT /user_attributes/;
      B26  7863              IFEND;
      B26  7864              IF shadow_sdtx_p^.shadow_info.shadow_segment_kind <> mmc$ssk_none THEN
      B32  7865                osp$set_status_abnormal ('MM', mme$invalid_shadow_segment, '', status);
      B62  7866                EXIT /user_attributes/;
      B66  7867              IFEND;
      B66  7868              sdtx_entry.shadow_info.shadow_segment_kind := mmc$ssk_segment_number;
      B66  7869              sdtx_entry.shadow_info.shadow_segment_number := #SEGMENT (attrib.user_attributes^ [i].shadow_p);
      B66  7870              sdtx_entry.shadow_info.shadow_sfid := shadow_sdtx_p^.sfid;
      B66  7871              shadow_sdtx_p^.shadow_info.passive_for_shadow_by_segnum := TRUE;
      B9C  7872            = mmc$kw_ps_transfer_size =
      B9C  7873              page_streaming_transfer_size := attrib.user_attributes^ [i].ps_transfer_size;
      B9C  7874              IF page_streaming_transfer_size > max_specified_transfer_size THEN
      BBA  7875                page_streaming_transfer_size := max_specified_transfer_size;
      BBE  7876              IFEND;
      BBE  7877              mmp$convert_ps_transfer_size (page_streaming_transfer_size, page_streaming_ts_shift);
      BF8  7878              sdtx_entry.stream.transfer_size := page_streaming_ts_shift;
      BF8  7879              sdtx_entry.stream.transfer_size_specified := TRUE;
      C12  7880            ELSE
```



```
      C12  7881                EXIT /user_attributes/;
      C16  7882              CASEND;
      C16  7883            FOREND;
      C1E  7884          IFEND;
      C1E  7885
      C1E  7886 {  Find an available segment number if the caller did not supply one.
      C1E  7887
      C1E  7888          pmp$find_executing_task_xcb (xcb_p);
      C24  7889
      C24  7890          IF segment_number = 0 THEN
      C40  7891            IF shared_taskid_array = NIL THEN
      C4E  7892              segment_res_state := mmc$srs_not_reserved;
      C54  7893            ELSE
      C54  7894              segment_res_state := mmc$srs_reserved_shared_stack;
      C56  7895            IFEND;
      C56  7896            find_available_segment_number (xcb_p, segment_res_state, segment_number, status);
      C78  7897          ELSE
      C78  7898            mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
      CA0  7899            IF NOT status.normal AND (status.condition = mme$segment_number_not_in_use) THEN
      CB8  7900              status.normal := TRUE;
      CB8  7901              WHILE (segment_number > xcb_p^.xp.segment_table_length) DO
      CD0  7902                expand_segment_table (xcb_p, status);
      CE2  7903                IF NOT status.normal THEN
      CEA  7904                  EXIT /user_attributes/;
      CF4  7905                IFEND;
      CF4  7906              WHILEND;
      D16  7907            ELSEIF status.normal THEN
      D1E  7908              osp$set_status_abnormal ('MM', mme$segment_number_is_in_use, '', status);
      D4E  7909            IFEND;
      D50  7910          IFEND;
      D50  7911        END /user_attributes/;
      D52  7912
      D52  7913        IF NOT status.normal THEN
      D5E  7914          gfp$unlock_fde_p (locked_fde_entry_p);
      E54  7915          IF attrib.sfid = gfv$null_sfid THEN
      E68  7916            gfp$free_fde (fde_entry_p);
      E7C  7917          IFEND;
      E7C  7918          RETURN;
      E7E  7919        IFEND;
      E7E  7920
      E7E  7921        locked_fde_entry_p^.last_segment_number := segment_number;
      E7E  7922        locked_fde_entry_p^.global_task_id := xcb_p^.global_task_id;
      E7E  7923
      E7E  7924        IF (mmc$sa_read_transfer_unit IN sdtx_entry.software_attribute_set) THEN
      EA2  7925          sdtx_entry.stream.sequential_accesses := mmv$page_streaming_prestream;
      EAE  7926        IFEND;
      EAE  7927
      EAE  7928        IF NOT sdtx_entry.stream.transfer_size_specified THEN
      EB8  7929          mmp$convert_ps_transfer_size (locked_fde_entry_p^.transfer_unit_size, page_streaming_ts_shift);
      EF8  7930          IF mmv$page_streaming_transfer > 0 THEN (override transfer size with mmv$page_streaming_transfer
      F04  7931            mmp$convert_ps_transfer_size (mmv$page_streaming_transfer, page_streaming_ts_shift);
      F3A  7932          IFEND;
      F3A  7933          sdtx_entry.stream.transfer_size := page_streaming_ts_shift;
      F4A  7934        IFEND;
      F4A  7935
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$BUILD_SEGMENT

```
F4A  7936      add_sdt_sdtx_entry (sdt_entry, sdtx_entry, locked_fde_entry_p, shared_taskid_array, segment_number);
F6A  7937
F6A  7938      IF locked_fde_entry_p^.file_limit < osc$maximum_offset THEN
F7A  7939        segment_length := locked_fde_entry_p^.file_limit;
F7E  7940      ELSE
F7E  7941        segment_length := osc$maximum_offset;
F84  7942      IFEND;
F84  7943
F84  7944      CASE attrib.pointer_kind OF
F98  7945      = mmc$sequence_pointer =
F98  7946        i#build_adaptable_seq_pointer (sdt_entry.ste.r1, segment_number, 0 (offset) , segment_length, 0,
FDO  7947              segment_pointer.seq_pointer);
FDO  7948      = mmc$heap_pointer =
FDO  7949        i#build_adaptable_heap_pointer (sdt_entry.ste.r1, segment_number, 0 (offset) , segment_length,
1004 7950              segment_pointer.heap_pointer);
1004 7951      ELSE
1004 7952        segment_pointer.cell_pointer := #ADDRESS (sdt_entry.ste.r1, segment_number, 0 (offset) );
1022 7953      CASEND;
1022 7954
1022 7955      gfp$unlock_fde_p (locked_fde_entry_p);
1116 7956
1116 7957    PROCEND mmp$build_segment;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$CHANGE_SEG_INHERITANCE_R1

```
 0  7959
 0  7960    PROCEDURE [XDCL, #GATE] mmp$change_seg_inheritance_r1
 0  7961      (     segment_number: ost$segment;
 0  7962            validating_ring: ost$valid_ring;
 0  7963            segment_inheritance: mmt$segment_inheritance;
 0  7964       VAR status: ost$status);
 0  7965
 0  7966      VAR
 0  7967        sdt_entry_p: ^mmt$segment_descriptor,
 0  7968        sdtx_entry_p: ^mmt$segment_descriptor_extended;
 0  7969
 0  7970      mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
30  7971      IF NOT status.normal THEN
38  7972        RETURN;
3A  7973      IFEND;
3A  7974
3A  7975      IF (validating_ring > sdt_entry_p^.ste.r2) THEN
4E  7976        osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
80  7977        RETURN;
82  7978      IFEND;
82  7979
82  7980      IF (segment_inheritance = mmc$si_transfer_segment) AND (sdtx_entry_p^.inheritance = mmc$si_none) THEN
98  7981        sdtx_entry_p^.inheritance := mmc$si_transfer_segment;
9E  7982      ELSEIF (segment_inheritance = mmc$si_share_segment) AND (sdtx_entry_p^.inheritance = mmc$si_none) THEN
B4  7983        sdtx_entry_p^.inheritance := mmc$si_share_segment;
BA  7984      ELSE
BA  7985        osp$set_status_abnormal ('MM', mme$illegal_segment_origin_chg, '', status);
EC  7986      IFEND;
EC  7987
EC  7988    PROCEND mmp$change_seg_inheritance_r1;
 0  7989
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$CHANGE_SEGMENT_NUMBER_R1

```
      0  7991
      0  7992   PROCEDURE [XDCL, #GATE] mmp$change_segment_number_r1
      0  7993     (     old_segment_number: ost$segment;
      0  7994           new_segment_number: ost$segment;
      0  7995           validating_ring_number: ost$valid_ring;
      0  7996      VAR status: ost$status);
      0  7997
      0  7998     VAR
      0  7999       fde_entry_p: gft$locked_file_desc_entry_p,
      0  8000       new_sdt_entry: mmt$segment_descriptor,
      0  8001       new_sdtx_entry: mmt$segment_descriptor_extended,
      0  8002       old_sdt_p: ^mmt$segment_descriptor,
      0  8003       old_sdtx_p: ^mmt$segment_descriptor_extended;
      0  8004
      0  8005     mmp$fetch_sdt_sdtx_locked_fde (old_segment_number, old_sdt_p, old_sdtx_p, fde_entry_p, status);
     38  8006     IF NOT status.normal THEN
     40  8007       RETURN;
     42  8008     IFEND;
     42  8009
     42  8010     IF (validating_ring_number > old_sdtx_p^.open_validating_ring_number) THEN
     52  8011       osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
     84  8012       RETURN;
     86  8013     IFEND;
     86  8014
     86  8015     new_sdt_entry := old_sdt_p^;
     86  8016     new_sdtx_entry := old_sdtx_p^;
     9C  8017
     9C  8018     add_sdt_sdtx_entry (new_sdt_entry, new_sdtx_entry, fde_entry_p, NIL, new_segment_number);
     C4  8019
     C4  8020     mmp$invalidate_segment (old_segment_number, 1, NIL (shared_taskid) , status);
     EE  8021
     EE  8022     gfp$unlock_fde_p (fde_entry_p);
    1E0  8023
    1E0  8024   PROCEND mmp$change_segment_number_r1;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$CHANGE_STACK_ATTRIBUTE_R1

```
      0  8026
      0  8027   PROCEDURE [XDCL, #GATE] mmp$change_stack_attribute_r1
      0  8028     (     stack_pages_to_be_freed: boolean;
      0  8029           caller_ring: ost$valid_ring;
      0  8030      VAR status: ost$status);
      0  8031
      0  8032     VAR
      0  8033       xcb_p: ^ost$execution_control_block;
      0  8034
      0  8035     status.normal := TRUE;
      4  8036     pmp$find_executing_task_xcb (xcb_p);
     14  8037     xcb_p^.stack_pages_saved [caller_ring] := NOT (stack_pages_to_be_freed);
     14  8038
     14  8039   PROCEND mmp$change_stack_attribute_r1;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$CLOSE_ASID_BASED_SEGMENT

```
      0  8041
      0  8042   PROCEDURE [XDCL] mmp$close_asid_based_segment
      0  8043     (     segment_number: ost$segment;
      0  8044      VAR status: ost$status);
      0  8045
      0  8046     VAR
      0  8047       pva: ^cell,
      0  8048       sdt_entry_p: ^mmt$segment_descriptor,
      0  8049       xcb_p: ^ost$execution_control_block;
      0  8050
      0  8051     status.normal := TRUE;
      4  8052     pmp$find_executing_task_xcb (xcb_p);
     14  8053
     14  8054     IF segment_number > xcb_p^.xp.segment_table_length THEN
     38  8055       osp$set_status_abnormal ('MM', mme$segment_number_too_big, '', status);
     6A  8056       RETURN;
     6C  8057     IFEND;
     6C  8058
     6C  8059     sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segment_number);
     6C  8060     IF sdt_entry_p^.ste.vl = osc$vl_invalid_entry THEN
     9E  8061       osp$set_status_abnormal ('MM', mme$segment_number_not_in_use, '', status);
     D0  8062       RETURN;
     D2  8063     IFEND;
     D2  8064
     D2  8065 {  Delete the Segment table entry.
     D2  8066
     D2  8067     pva := #ADDRESS (1, segment_number, 0);
     D2  8068     #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
     E6  8069     sdt_entry_p^.ste.vl := osc$vl_invalid_entry;
     E6  8070
     E6  8071   PROCEND mmp$close_asid_based_segment;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$CLOSE_DEVICE_FILE

```
      0  8073
      0  8074   PROCEDURE [XDCL, #GATE] mmp$close_device_file
      0  8075     (     segment_number: ost$segment;
      0  8076      VAR status: ost$status);
      0  8077
      0  8078     VAR
      0  8079       fde_entry_p: gft$locked_file_desc_entry_p,
      0  8080       pva: ^cell,
      0  8081       sdt_entry_p: ^mmt$segment_descriptor,
      0  8082       sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  8083
      0  8084     status.normal := TRUE;
      4  8085
      4  8086     mmp$fetch_sdt_sdtx_locked_fde (segment_number, sdt_entry_p, sdtx_entry_p, fde_entry_p, status);
     3E  8087
     3E  8088 {  Delete the Segment table entry.
     3E  8089
     3E  8090     pva := #ADDRESS (1, segment_number, 0);
     3E  8091     #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
     4E  8092     sdt_entry_p^.ste.vl := osc$vl_invalid_entry;
     4E  8093     fde_entry_p^.open_count := fde_entry_p^.open_count - 1;
     4E  8094     gfp$unlock_fde_p (fde_entry_p);
    160  8095
    160  8096   PROCEND mmp$close_device_file;
```

```
      0  8098 {
      0  8099 {   The purpose of this request is to create the SDT and SDTX for a
      0  8100 { new task. All valid segments with an SDTX.INHERITANCE of mmc$si_share_segment
      0  8101 { in the parent's SDTX are inherited by the new task. For segments in the parent's
      0  8102 { SDTX with an inheritance of mmc$si_new_segment, a new segment is created in the
      0  8103 { new task (these are primarily the task template segments). All segments in the
      0  8104 { parent's SDTX with a segment reservation of mmc$srs_reserved_shared_stack
      0  8105 { are inherited, regardless, of whether or not the segment is valid.
      0  8106 {
      0  8107 {   The ASID and ASTI of inherited segments are zeroed out. These values
      0  8108 { may be copied into the child's segment table during a later phase of
      0  8109 { task initiation, in the procedure MMP$CREATE_TASK.
      0  8110 {
      0  8111 {       MMP$CREATE_INHERITED_SDT (TASK_ID, STATUS)
      0  8112 {
      0  8113 { TASK_ID: (input) This parameter identifies the task being created.
      0  8114 {
      0  8115 { STATUS: (output) The request status is returned in this parameter.
      0  8116 {       The possible error codes are:
      0  8117 {             mme$invalid_task_id
      0  8118 {             mme$sdt_or_sdtx_exists
      0  8119 {
      0  8120
      0  8121   PROCEDURE [XDCL, #GATE] mmp$create_inherited_sdt
      0  8122     (    task_id: pmt$task_id;
      0  8123      VAR status: ost$status);
      0  8124
      0  8125     VAR
      0  8126       caller_id: ost$caller_identifier,
      0  8127       child_sdt_entry_p: ^mmt$segment_descriptor,
      0  8128       child_sdtx_entry_p: ^mmt$segment_descriptor_extended,
      0  8129       child_xcb_p (new task xcb pointer) : ^ost$execution_control_block,
      0  8130       fde_entry_p: gft$locked_file_desc_entry_p,
      0  8131       local_sdtp: ^mmt$segment_descriptor_table,
      0  8132       local_sdtxp: ^mmt$segment_descriptor_table_ex,
      0  8133       local_status: ost$status,
      0  8134       new_sdt_length: integer,
      0  8135       new_table_size: ost$segment_length,
      0  8136       parent_sdt_p: mmt$max_sdt_p,
      0  8137       parent_sdtx_p: mmt$max_sdtx_p,
      0  8138       parent_xcb_p (current task xcb pointer) : ^ost$execution_control_block,
      0  8139       pva: ^cell,
      0  8140       rma: integer,
      0  8141       sdt_entry: mmt$segment_descriptor,
      0  8142       sdtx_entry: mmt$segment_descriptor_extended,
      0  8143       segnum: ost$segment,
      0  8144       sfid: gft$system_file_identifier,
      0  8145       software_attribute_set: mmt$software_attribute_set,
      0  8146       st1 (segment table length) : ost$segment;
      0  8147
      0  8148     status.normal := TRUE;
      4  8149
      4  8150     pmp$find_task_xcb (task_id, child_xcb_p);
     2A  8151     IF child_xcb_p = NIL THEN
     38  8152       osp$set_status_abnormal ('MM', mme$invalid_task_id, '', status);
```



```
     6A  8153       RETURN;
     6C  8154     IFEND;
     6C  8155
     6C  8156     pmp$find_executing_task_xcb (parent_xcb_p);
     72  8157
     72  8158     st1 := parent_xcb_p^.xp.segment_table_length;
     72  8159     mmp$get_max_sdt_sdtx_pointer (parent_xcb_p, parent_sdt_p, parent_sdtx_p);
     72  8160     WHILE ((parent_sdtx_p^.sdtx_table [st1].inheritance = mmc$si_none) OR
    104  8161         (parent_sdt_p^.st [st1].ste.v1 = osc$v1_invalid_entry)) AND
    104  8162         (parent_sdtx_p^.sdtx_table [st1].segment_reservation_state <> mmc$srs_reserved_shared_stack) AND
    104  8163         (st1 > mmc$default_sdt_length) DO
    104  8164       st1 := st1 - 1;
    104  8165     WHILEND;
    14E  8166
    14E  8167     ALLOCATE local_sdtp: [0 .. st1] IN osv$job_fixed_heap^;
    18C  8168
    18C  8169     IF ((st1 + 1) * 8) > osv$page_size THEN
    198  8170       new_sdt_length := ((((st1 + 1) * 8) + osv$page_size) DIV osv$page_size);
    198  8171       new_table_size := new_sdt_length * osv$page_size;
    1A4  8172     IFEND;
    1A4  8173
    1A4  8174 {  Allocate and initialize the SDT.
    1A4  8175     IF ((st1 + 1) * 8) > osv$page_size THEN
    1B2  8176       mmp$free_pages (local_sdtp, new_table_size, osc$nowait, status);
    1D6  8177       mmp$assign_contiguous_memory (local_sdtp, new_table_size, status);
    1F2  8178       IF NOT status.normal THEN
    1FA  8179         RETURN;
    1FC  8180       IFEND;
    200  8181     ELSE
    200  8182       pmp$zero_out_table (#LOC (local_sdtp^), #SIZE (local_sdtp^));
    21C  8183     IFEND;
    21C  8184
    21C  8185
    21C  8186 {  Allocate and zero out the SDTX.
    21C  8187
    21C  8188     ALLOCATE local_sdtxp: [0 .. st1] IN osv$job_fixed_heap^;
    25C  8189     pmp$zero_out_table (#LOC (local_sdtxp^), #SIZE (local_sdtxp^));
    278  8190     i#real_memory_address (local_sdtp, rma);
    294  8191     child_xcb_p^.xp.segment_table_address_1 := rma DIV 10000(16);
    294  8192     child_xcb_p^.xp.segment_table_address_2 := rma MOD 10000(16);
    294  8193     child_xcb_p^.xp.segment_table_length := st1;
    294  8194     child_xcb_p^.sdt_offset := #OFFSET (local_sdtp);
    294  8195     child_xcb_p^.sdtx_offset := #OFFSET (local_sdtxp);
    294  8196
    294  8197 {  Create the SDT and SDTX in the new task.
    294  8198
    294  8199   /create_sdt_and_sdtx/
    294  8200     FOR segnum := 0 TO st1 DO
    2E6  8201       IF (parent_sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND
    31A  8202             ((parent_sdtx_p^.sdtx_table [segnum].inheritance <> mmc$si_none) OR
    31A  8203             (parent_sdtx_p^.sdtx_table [segnum].segment_reservation_state =
    31A  8204           mmc$srs_reserved_shared_stack)) THEN
    31A  8205         sdt_entry := parent_sdt_p^.st [segnum];
    31A  8206         sdt_entry.ste.asid := 0;
    31A  8207         sdt_entry.asti := 0;
```

```
     31A  8208          sdtx_entry := parent_sdtx_p^.sdtx_table [segnum];
     348  8209          IF (sdtx_entry.shadow_info.shadow_segment_kind <> mmc$ssk_none) AND
     356  8210                (sdtx_entry.shadow_info.shadow_segment_kind <> mmc$ssk_segment_number) THEN
     356  8211            sdtx_entry.shadow_info.shadow_segment_kind := mmc$ssk_none;
     356  8212            sdtx_entry.shadow_info.passive_for_shadow_by_segnum := FALSE;
     356  8213            sdtx_entry.sfid := sdtx_entry.shadow_info.shadow_sfid;
     366  8214          IFEND;
     366  8215
     366  8216          sdtx_entry.assign_active := mmc$assign_active_null;
     366  8217
     366  8218 { New FDE entries must be created for the task template segments.
     366  8219
     366  8220          IF sdtx_entry.inheritance = mmc$si_new_segment THEN
     378  8221            gfp$assign_fde (gfc$tr_job, 0 (segment_number) , sfid, fde_entry_p);
     39E  8222            IF fde_entry_p <> NIL THEN
     3A8  8223              sfid.file_hash := segnum;
     3A8  8224              fde_entry_p^.open_count := 1;
     3A8  8225              fde_entry_p^.attach_count := 1;
     3A8  8226              fde_entry_p^.file_kind := gfc$fk_unnamed_file;
     3A8  8227              fde_entry_p^.file_hash := segnum;
     3A8  8228              fde_entry_p^.last_segment_number := segnum;
     3A8  8229              IF mmc$sa_stack IN sdtx_entry.software_attribute_set THEN
     3CE  8230                fde_entry_p^.stack_for_ring := sdt_entry.ste.r1;
     3D6  8231              IFEND;
     3D6  8232              sdtx_entry.sfid := sfid;
     3E2  8233            ELSE
     3E2  8234              osp$set_status_abnormal ('MM', mme$unable_to_assign_fde, '', status);
     412  8235              RETURN;
     414  8236            IFEND;
     418  8237          ELSEIF parent_sdtx_p^.sdtx_table [segnum].inheritance = mmc$si_transfer_segment THEN
     430  8238            sdtx_entry.inheritance := mmc$si_none;
     430  8239            pva := #ADDRESS (1, segnum, 0);
     430  8240            #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
     442  8241            parent_sdt_p^.st [segnum].ste.v1 := osc$v1_invalid_entry;
     45C  8242          ELSEIF sdtx_entry.open_validating_ring_number > 1 THEN
     464  8243            gfp$get_locked_fde_p (sdtx_entry.sfid, fde_entry_p);
     58E  8244            fde_entry_p^.open_count := fde_entry_p^.open_count + 1;
     58E  8245            gfp$unlock_fde_p (fde_entry_p);
     68E  8246            sdtx_entry.software_attribute_set := sdtx_entry.software_attribute_set -
     69E  8247                  $mmt$software_attribute_set [mmc$sa_stack];
     69E  8248          IFEND;
     69E  8249          child_sdt_entry_p := mmp$get_sdt_entry_p (child_xcb_p, segnum);
     69E  8250          child_sdtx_entry_p := mmp$get_sdtx_entry_p (child_xcb_p, segnum);
     69E  8251          child_sdt_entry_p^ := sdt_entry;
     69E  8252          child_sdtx_entry_p^ := sdtx_entry;
     6FC  8253
     6FC  8254        ELSEIF (parent_sdtx_p^.sdtx_table [segnum].segment_reservation_state =
     728  8255              mmc$srs_reserved_shared_stack) AND
     728  8256              (parent_sdt_p^.st [segnum].ste.v1 = osc$v1_invalid_entry) THEN
     728  8257            sdtx_entry := parent_sdtx_p^.sdtx_table [segnum];
     734  8258            sdtx_entry.segment_reservation_state := mmc$srs_reserved_shared_stack;
     734  8259            child_sdtx_entry_p := mmp$get_sdtx_entry_p (child_xcb_p, segnum);
     734  8260            child_sdtx_entry_p^ := sdtx_entry;
     76E  8261        IFEND;
     76E  8262      FOREND /create_sdt_and_sdtx/;

                    *
```



```
     772  8263
     772  8264    PROCEND mmp$create_inherited_sdt;
       0  8265
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$DELETE_NON_INHERITED_SEGS

```
0  8267 {
0  8268 {   The purpose of this request is to close all segments in the task's SDT
0  8269 { that do not have the inherited attribute. Segments which have a reservation
0  8270 { state of mmc$srs_reserved_shared_stack are not closed.
0  8271 {
0  8272 {       MMP$DELETE_NON_INHERITED_SEGS (STATUS)
0  8273 {
0  8274 { STATUS: (output) The request status is returned in this parameter.
0  8275 {         The possible error codes are:
0  8276 {               mme$invalid_close_segment_req
0  8277 {               mme$invalid_shared_taskid
0  8278 {               mme$segment_number_not_in_use
0  8279 {               mme$segment_number_too_big
0  8280 {
0  8281
0  8282   PROCEDURE [XDCL, #GATE] mmp$delete_non_inherited_segs
0  8283     (VAR status: ost$status);
0  8284
0  8285     VAR
0  8286       pointer: mmt$segment_pointer,
0  8287       sdt_p: mmt$max_sdt_p,
0  8288       sdtx_p: mmt$max_sdtx_p,
0  8289       segnum: ost$segment,
0  8290       xcb_p: ^ost$execution_control_block;
0  8291
0  8292     status.normal := TRUE;
4  8293     pmp$find_executing_task_xcb (xcb_p);
14 8294
14 8295 {  Close all user segments.
14 8296
14 8297     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
14 8298     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
6A 8299       IF ((sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND
94 8300           (sdtx_p^.sdtx_table [segnum].open_validating_ring_number > 1)) THEN
94 8301         mmp$invalidate_segment (segnum, 1, NIL (shared_taskid_array) , status);
BC 8302       IFEND;
BC 8303     FOREND;
CO 8304
CO 8305   PROCEND mmp$delete_non_inherited_segs;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$FETCH_OFFSET_MOD_PAGES_R1

```
0  8307
0  8308   PROCEDURE [XDCL, #GATE] mmp$fetch_offset_mod_pages_r1
0  8309     (     segment_number: ost$segment;
0  8310           xsfid: gft$system_file_identifier;
0  8311           return_unallocated_offsets: boolean;
0  8312      VAR offset_list: ^array [ * ] of ost$segment_offset;
0  8313      VAR offsets_returned: integer;
0  8314      VAR status: ost$status);
0  8315
0  8316     TYPE
0  8317       offset_array = array [1 .. * ] of ost$segment_offset;
0  8318
0  8319     VAR
0  8320       i: integer,
0  8321       offset_array_index: integer,
0  8322       offset_count: integer,
0  8323       offset_p: ^offset_array,
0  8324       request_block: mmt$rb_fetch_offset_mod_pages,
0  8325       sdt_entry_p: ^mmt$segment_descriptor,
0  8326       sdtx_entry_p: ^mmt$segment_descriptor_extended,
0  8327       sfid: gft$system_file_identifier;
0  8328
0  8329     status.normal := TRUE;
4  8330
4  8331 {  Allocate array to hold all the offsets for the modified pages.
4  8332
4  8333     ALLOCATE offset_p: [1 .. UPPERBOUND (offset_list^)] IN osv$job_fixed_heap^;
66 8334
66 8335 {  Touch all of the pages allocated so that referencing them in
66 8336 {  monitor will not cause a page fault.
66 8337
66 8338     mmp$touch_all_pages (offset_p, #SIZE (offset_p^));
98 8339
98 8340 {  Issue monitor function to return offsets for modified pages.
98 8341
98 8342 { If the segment number is non-zero, then the sfid passed into the request
98 8343 { is invalid. The correct sfid must be set from the segment's SDTX entry.
98 8344
98 8345     IF segment_number <> 0 THEN
A6 8346       mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
CA 8347       IF NOT status.normal THEN
D2 8348         RETURN;
D4 8349       IFEND;
D4 8350       sfid := sdtx_entry_p^.sfid;
EO 8351     ELSE
EO 8352       sfid := xsfid;
E4 8353     IFEND;
E4 8354
E4 8355     request_block.reqcode := syc$rc_fetch_offset_mod_pages;
E4 8356     request_block.sfid := sfid;
E4 8357     request_block.offsets_returned := UPPERBOUND (offset_list^);
E4 8358     request_block.offset_list := offset_p;
112 8359    request_block.return_unallocated_offsets := return_unallocated_offsets;
112 8360
112 8361    i#call_monitor (#LOC (request_block), #SIZE (request_block));
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$FETCH_OFFSET_MOD_PAGES_R1

```
  136  8362      syp$set_status_from_mtr_status (request_block.status, status);
  172  8363      IF NOT status.normal THEN
  17A  8364        FREE offset_p IN osv$job_fixed_heap^;
  1A0  8365        RETURN;
  1AC  8366      IFEND;
  1AC  8367
  1AC  8368 {  If array not large enough to hold all offsets, create list_overflow condition.
  1AC  8369
  1AC  8370      IF (request_block.offsets_returned > UPPERBOUND (offset_list^)) OR
  1CC  8371           (request_block.offsets_returned = 0) THEN
  1CC  8372        FREE offset_p IN osv$job_fixed_heap^;
  1F2  8373        offsets_returned := request_block.offsets_returned;
  1F2  8374        RETURN;
  20A  8375      IFEND;
  20A  8376
  20A  8377 {  Move offsets to caller's offset list.
  20A  8378
  20A  8379      FOR i := 1 TO request_block.offsets_returned DO
  214  8380        offset_list^ [i] := offset_p^ [i];
  214  8381      FOREND;
  23C  8382
  23C  8383      offsets_returned := request_block.offsets_returned;
  23C  8384      FREE offset_p IN osv$job_fixed_heap^;
  26E  8385
  26E  8386    PROCEND mmp$fetch_offset_mod_pages_r1;
    0  8387
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$FETCH_SDT_SDTX_LOCKED_FDE

```
    0  8389
    0  8390    PROCEDURE [XDCL, #GATE] mmp$fetch_sdt_sdtx_locked_fde
    0  8391      (     segment_number: ost$segment;
    0  8392       VAR sdt_entry_p: ^mmt$segment_descriptor;
    0  8393       VAR sdtx_entry_p: ^mmt$segment_descriptor_extended;
    0  8394       VAR locked_fde_p: gft$locked_file_desc_entry_p;
    0  8395       VAR status: ost$status);
    0  8396
    0  8397      VAR
    0  8398        xcb_p: ^ost$execution_control_block;
    0  8399
    0  8400      status.normal := TRUE;
    4  8401      pmp$find_executing_task_xcb (xcb_p);
   14  8402
   14  8403      IF segment_number > xcb_p^.xp.segment_table_length THEN
   38  8404        IF segment_number > 4095 THEN
   40  8405          osp$set_status_abnormal ('MM', mme$segment_number_too_big, '', status);
   74  8406        ELSE
   74  8407          osp$set_status_abnormal ('MM', mme$segment_number_not_in_use, '', status);
   A6  8408        IFEND;
   A6  8409        RETURN;
   A8  8410      IFEND;
   A8  8411
   A8  8412      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segment_number);
   A8  8413      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segment_number);
   A8  8414      IF sdt_entry_p^.ste.v1 = osc$v1_invalid_entry THEN
  116  8415        osp$set_status_abnormal ('MM', mme$segment_number_not_in_use, '', status);
  14A  8416      ELSE
  14A  8417        gfp$get_locked_fde_p (sdtx_entry_p^.sfid, locked_fde_p);
  27A  8418      IFEND;
  27A  8419
  27A  8420    PROCEND mmp$fetch_sdt_sdtx_locked_fde;
    0  8421
```

```
      0  8423
      0  8424    PROCEDURE [XDCL, #GATE] mmp$fetch_segment_attributes_r1
      0  8425      (     segment_number: ost$segment;
      0  8426            validating_ring: ost$valid_ring;
      0  8427       VAR seg_attributes: ^array [ * ] of mmt$attribute_descriptor;
      0  8428       VAR status: ost$status);
      0  8429
      0  8430      VAR
      0  8431        fde_entry_p: gft$locked_file_desc_entry_p,
      0  8432        i: integer,
      0  8433        sdt_entry_p: ^mmt$segment_descriptor,
      0  8434        sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  8435
      0  8436      status.normal := TRUE;
      4  8437      mmp$fetch_sdt_sdtx_locked_fde (segment_number, sdt_entry_p, sdtx_entry_p, fde_entry_p, status);
     3E  8438      FOR i := LOWERBOUND (seg_attributes^) TO UPPERBOUND (seg_attributes^) DO
     5C  8439        CASE seg_attributes^ [i].keyword OF
     DA  8440        = mmc$kw_ring_numbers =
     DA  8441          seg_attributes^ [i].r1 := sdt_entry_p^.ste.r1;
     DA  8442          seg_attributes^ [i].r2 := sdt_entry_p^.ste.r2;
    10C  8443        = mmc$kw_segment_number =
    10C  8444          seg_attributes^ [i].segnum := segment_number;
    118  8445        = mmc$kw_current_segment_length =
    118  8446          seg_attributes^ [i].current_length := gfp$get_eoi_from_fde (fde_entry_p);
    15C  8447        = mmc$kw_max_segment_length =
    15C  8448          IF fde_entry_p^.file_limit < UPPERVALUE (seg_attributes^ [i].max_length) THEN
    17E  8449            seg_attributes^ [i].max_length := fde_entry_p^.file_limit;
    186  8450          ELSE
    186  8451            seg_attributes^ [i].max_length := UPPERVALUE (seg_attributes^ [i].max_length);
    18E  8452          IFEND;
    192  8453        = mmc$kw_gl_key =
    192  8454          seg_attributes^ [i].gl_key := sdt_entry_p^.ste.key_lock;
    1B4  8455        = mmc$kw_hardware_attributes =
    1B4  8456          seg_attributes^ [i].hardware_attri_set := $mmt$hardware_attribute_set [];
    1B4  8457          CASE sdt_entry_p^.ste.rp OF
    1EC  8458          = osc$read_uncontrolled =
    1EC  8459            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    210  8460                  $mmt$hardware_attribute_set [mmc$ha_read];
    210  8461          = osc$read_key_lock_controlled =
    210  8462            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    220  8463                  $mmt$hardware_attribute_set [mmc$ha_read_key_lock];
    220  8464          = osc$binding_segment =
    220  8465            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    22E  8466                  $mmt$hardware_attribute_set [mmc$ha_binding]
    22E  8467          ELSE
    22E  8468          CASEND;
    22E  8469          CASE sdt_entry_p^.ste.wp OF
    246  8470          = osc$write_uncontrolled =
    246  8471            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    26A  8472                  $mmt$hardware_attribute_set [mmc$ha_write];
    26A  8473          = osc$write_key_lock_controlled =
    26A  8474            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    28E  8475                  $mmt$hardware_attribute_set [mmc$ha_write_key_lock];
    28E  8476          ELSE
    28E  8477          CASEND;
```



```
    28E  8478          CASE sdt_entry_p^.ste.xp OF
    2AC  8479          = osc$non_privileged =
    2AC  8480            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    2CE  8481                  $mmt$hardware_attribute_set [mmc$ha_execute];
    2CE  8482          = osc$local_privilege =
    2CE  8483            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    2F0  8484                  $mmt$hardware_attribute_set [mmc$ha_execute_local];
    2F0  8485          = osc$global_privilege =
    2F0  8486            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    310  8487                  $mmt$hardware_attribute_set [mmc$ha_execute_global];
    310  8488          ELSE
    310  8489          CASEND;
    310  8490          CASE sdt_entry_p^.ste.v1 OF
    322  8491          = osc$v1_cache_bypass =
    322  8492            seg_attributes^ [i].hardware_attri_set := seg_attributes^ [i].hardware_attri_set +
    342  8493                  $mmt$hardware_attribute_set [mmc$ha_cache_bypass];
    342  8494          ELSE
    342  8495          CASEND;
    346  8496        = mmc$kw_software_attributes =
    346  8497          seg_attributes^ [i].software_attri_set := sdtx_entry_p^.software_attribute_set;
    368  8498        = mmc$kw_error_exit_procedure =
    368  8499          seg_attributes^ [i].err_exit_proc := NIL;
    38E  8500        = mmc$kw_preset_value =
    38E  8501          seg_attributes^ [i].preset_value := fde_entry_p^.preset_value;
    3B0  8502        = mmc$kw_inheritance =
    3B0  8503          seg_attributes^ [i].inheritance := sdtx_entry_p^.inheritance;
    3D2  8504        = mmc$kw_clear_space =
    3D2  8505          seg_attributes^ [i].clear_space := FALSE;
    3EE  8506        = mmc$kw_segment_access_control =
    3EE  8507          IF sdt_entry_p^.ste.v1 = osc$v1_cache_bypass THEN
    400  8508            seg_attributes^ [i].access_control.cache_bypass := TRUE;
    41A  8509          ELSE
    41A  8510            seg_attributes^ [i].access_control.cache_bypass := FALSE;
    432  8511          IFEND;
    432  8512          seg_attributes^ [i].access_control.execute_privilege := sdt_entry_p^.ste.xp;
    432  8513          seg_attributes^ [i].access_control.read_privilege := sdt_entry_p^.ste.rp;
    432  8514          seg_attributes^ [i].access_control.write_privilege := sdt_entry_p^.ste.wp;
    46A  8515        ELSE
    46A  8516          osp$set_status_abnormal ('MM', mme$unsupported_keyword, '', status);
    49C  8517        CASEND;
    49C  8518      FOREND;
    4A0  8519
    4A0  8520      gfp$unlock_fde_p (fde_entry_p);
    596  8521
    596  8522    PROCEND mmp$fetch_segment_attributes_r1;
      0  8523
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$GET_ALLOCATED_ADDRESSES_R1

```
  0  8525
  0  8526   PROCEDURE [XDCL, #GATE] mmp$get_allocated_addresses_r1
  0  8527     (     segment_number: ost$segment;
  0  8528           validating_ring: ost$valid_ring;
  0  8529           starting_byte_address: ost$segment_offset;
  0  8530      VAR addr_list: ^array [ * ] of dmt$addr_length_pair;
  0  8531      VAR addr_returned: integer;
  0  8532      VAR list_overflow: boolean;
  0  8533      VAR status: ost$status);
  0  8534
  0  8535     VAR
  0  8536       allocated_length: amt$file_byte_address,
  0  8537       dfd_p: ^dmt$disk_file_descriptor,
  0  8538       fde_entry_p: gft$locked_file_desc_entry_p,
  0  8539       sdt_entry_p: ^mmt$segment_descriptor,
  0  8540       sdtx_entry_p: ^mmt$segment_descriptor_extended;
  0  8541
  0  8542     mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
 30  8543     IF NOT status.normal THEN
 38  8544       RETURN;
 3A  8545     IFEND;
 3A  8546     IF (sdt_entry_p^.ste.rp = osc$non_readable) OR (validating_ring > sdt_entry_p^.ste.r2) THEN
 5A  8547       osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
 8C  8548       RETURN;
 8E  8549     IFEND;
 8E  8550
 8E  8551     IF starting_byte_address = 0 THEN
 96  8552       gfp$get_locked_fde_p (sdtx_entry_p^.sfid, fde_entry_p);
1D0  8553       dmp$get_disk_file_descriptor_p (fde_entry_p, dfd_p);
1E4  8554       dmp$get_total_allocated_length (fde_entry_p, allocated_length);
214  8555       gfp$unlock_fde_p (fde_entry_p);
308  8556       IF allocated_length = dfd_p^.highest_offset_allocated THEN
318  8557         {Return 1 address pair if not sparsely allocated
318  8558         addr_returned := 1;
318  8559         list_overflow := FALSE;
318  8560         addr_list^ [LOWERBOUND (addr_list^)].addr := 0;
318  8561         addr_list^ [LOWERBOUND (addr_list^)].length := fde_entry_p^.eoi_byte_address;
318  8562         RETURN;
35E  8563       IFEND;
35E  8564     IFEND;
35E  8565
35E  8566     dmp$get_initialized_addresses (sdtx_entry_p^.sfid, starting_byte_address, addr_list^, addr_returned,
3AC  8567           list_overflow, status);
3AC  8568
3AC  8569   PROCEND mmp$get_allocated_addresses_r1;
  0  8570
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$GET_SDT_FOR_JOB_TEMPLATE

```
  0  8572
  0  8573   PROCEDURE [XDCL] mmp$get_sdt_for_job_template
  0  8574     (     pva: ^cell;
  0  8575      VAR sdt_entry: mmt$segment_descriptor;
  0  8576      VAR sdtx_entry: mmt$segment_descriptor_extended;
  0  8577      VAR status: ost$status);
  0  8578
  0  8579
  0  8580     VAR
  0  8581       fde_entry_p: gft$locked_file_desc_entry_p,
  0  8582       sdt_entry_p: ^mmt$segment_descriptor,
  0  8583       sdtx_entry_p: ^mmt$segment_descriptor_extended,
  0  8584       segnum: ost$segment;
  0  8585
  0  8586     segnum := #SEGMENT (pva);
  4  8587     mmp$fetch_sdt_sdtx_locked_fde (segnum, sdt_entry_p, sdtx_entry_p, fde_entry_p, status);
 3E  8588     IF status.normal = FALSE THEN
 46  8589       RETURN;
 48  8590     IFEND;
 48  8591
 48  8592     fde_entry_p^.flags.global_template_file := TRUE;
 48  8593     sdtx_entry_p^.inheritance := mmc$si_share_segment;
 48  8594     sdtx_entry_p^.open_validating_ring_number := 0;
 48  8595     sdtx_entry_p^.file_limits_enforced := sfc$no_limit;
 48  8596     sdtx_entry := sdtx_entry_p^;
 74  8597     sdt_entry := sdt_entry_p^;
 74  8598     sdt_entry.ste.asid := 0;
 74  8599     gfp$unlock_fde_p (fde_entry_p);
176  8600
176  8601   PROCEND mmp$get_sdt_for_job_template;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$GET_SEGMENT_LENGTH_R1

```
      0  8603
      0  8604    PROCEDURE [XDCL, #GATE] mmp$get_segment_length_r1
      0  8605      (     segment_number: ost$segment;
      0  8606            validating_ring_number: ost$valid_ring;
      0  8607       VAR segment_length: ost$segment_length;
      0  8608       VAR status: ost$status);
      0  8609
      0  8610      VAR
      0  8611        fde_p: gft$file_desc_entry_p,
      0  8612        sdt_entry_p: ^mmt$segment_descriptor,
      0  8613        sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  8614
      0  8615      mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
     30  8616      IF status.normal = FALSE THEN
     38  8617        RETURN;
     3A  8618      IFEND;
     3A  8619
     3A  8620 {  Verify that caller is within read bracket of the segment.
     3A  8621
     3A  8622      IF validating_ring_number > sdt_entry_p^.ste.r2 THEN
     4E  8623        osp$set_status_abnormal ('MM', mme$caller_not_in_read_bracket, '', status);
     80  8624        RETURN;
     82  8625      IFEND;
     82  8626
     82  8627      gfp$get_fde_p (sdtx_entry_p^.sfid, fde_p);
     DE  8628      segment_length := gfp$get_eoi_from_fde (fde_p);
    118  8629
    118  8630    PROCEND mmp$get_segment_length_r1;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$INITIATE_SHADOWING_R1

```
      0  8632 {
      0  8633 {    The purpose of this request is to execute ring 1 code for MMP$INITIATE_SHADOWING.
      0  8634 {    Segment attributes are updated to transform that segment into an ACTIVE (shadow)
      0  8635 {    segment.
      0  8636 {
      0  8637 {         MMP$INITIATE_SHADOWING_R1 (SEGMENT_POINTER, SEGMENT_LENGTH, STATUS);
      0  8638 {
      0  8639 { SEGMENT_POINTER: (input) This parameter specifies the process virtual
      0  8640 {         address assigned to the segment.
      0  8641 {
      0  8642 { SEGMENT_LENGTH: (input) This parameter specifies the segment length
      0  8643 {         for the segment that is to be shadowed.
      0  8644 {
      0  8645 { STATUS: (output) This parameter specifies the request status.
      0  8646 {
      0  8647
      0  8648
      0  8649    PROCEDURE [XDCL, #GATE] mmp$initiate_shadowing_r1
      0  8650      (     segment_pointer: ^cell;
      0  8651            validating_ring_number: ost$valid_ring;
      0  8652            shadow_segment_kind: mmt$shadow_segment_kind;
      0  8653       VAR status: ost$status);
      0  8654
      0  8655      VAR
      0  8656        fde_p: gft$locked_file_desc_entry_p,
      0  8657        new_fde_p: gft$file_desc_entry_p,
      0  8658        new_sfid: gft$system_file_identifier,
      0  8659        segment_length: ost$segment_length,
      0  8660        sdt_entry_p: ^mmt$segment_descriptor,
      0  8661        sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  8662
      0  8663
      0  8664      mmp$fetch_sdt_sdtx_locked_fde (#SEGMENT (segment_pointer), sdt_entry_p, sdtx_entry_p, fde_p, status);
     3E  8665      IF NOT status.normal THEN
     46  8666        RETURN;
     48  8667      IFEND;
     48  8668
     48  8669 {!!!### why was there a check for R1 = R2 ???}
     48  8670 {!!!### when ADA task crates a child task, what happens to debug-shadowed code segments??
     48  8671
     48  8672    /fde_locked/
     48  8673      BEGIN
     48  8674        IF (sdt_entry_p^.ste.r1 < validating_ring_number) OR (sdtx_entry_p^.open_validating_ring_number <= 1) OR
     76  8675              (mmc$sa_stack IN sdtx_entry_p^.software_attribute_set) THEN
     76  8676          osp$set_status_abnormal ('MM', mme$init_shadow_improper_seg, '', status);
     A8  8677          EXIT /fde_locked/;
     AC  8678        IFEND;
     AC  8679
     AC  8680        gfp$assign_fde (gfc$tr_job, 0, new_sfid, new_fde_p);
     D4  8681        IF new_fde_p = NIL THEN
     E2  8682          osp$set_status_abnormal ('MM', mme$unable_to_assign_fde, '', status);
    112  8683          EXIT /fde_locked/;
    118  8684        IFEND;
    118  8685        new_fde_p^.allocation_unit_size := mmc$shadow_allocation_size;
    118  8686        new_fde_p^.open_count := 1;
```

```
118  8687        new_fde_p^.attach_count := 1;
118  8688
118  8689        IF shadow_segment_kind = mmc$ssk_read_only_file THEN
130  8690          sdt_entry_p^.ste.wp := osc$write_uncontrolled;
130  8691          sdtx_entry_p^.access_rights := mmc$sar_write_extend;
14A  8692        IFEND;
14A  8693
14A  8694        mmp$get_segment_length_r1 (#SEGMENT (segment_pointer), 1, segment_length, status);
174  8695
174  8696        sdtx_entry_p^.shadow_info.shadow_segment_kind := shadow_segment_kind;
174  8697        sdtx_entry_p^.shadow_info.shadow_sfid := sdtx_entry_p^.sfid;
174  8698        sdtx_entry_p^.shadow_info.shadow_start_page_number := 0;
174  8699        sdtx_entry_p^.shadow_info.shadow_length_page_count := (((segment_length + 16384 - 1) DIV 16384) *
174  8700             16384) DIV osv$page_size;
174  8701        new_fde_p^.file_limit := fde_p^.file_limit;
174  8702        sdtx_entry_p^.sfid := new_sfid;
174  8703
174  8704        {  Purge buffer space for PASSIVE segment and set ASID to zero.
174  8705
174  8706        sdt_entry_p^.ste.asid := 0;
174  8707        #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, segment_pointer);
1CE  8708
1CE  8709      END /fde_locked/;
1DO  8710
1DO  8711      gfp$unlock_fde_p (fde_p);
2C6  8712
2C6  8713    PROCEND mmp$initiate_shadowing_r1;
```



```
  0  8715 {
  0  8716 {   This procedure initializes the system privilege bit map.  It first
  0  8717 { clears the map, located in the mainframe pageable segment, then decides
  0  8718 { which segments have system privilege, and sets the corresponding bits
  0  8719 { in the map.  Before this procedure is called, the bit map must have
  0  8720 { been moved from the task private segment static area into mainframe
  0  8721 { pageable.  After this procedure is called, each new task will pick up
  0  8722 { the initialized map from mainframe pageable into its task private segment.
  0  8723 {
  0  8724 {    MMP$INIT_SYSTEM_PRIVILEGE_MAP (offset);
  0  8725 {
  0  8726 { OFFSET: (input) This parameter specifies the offset of the privilege map.
  0  8727 {
  0  8728
  0  8729   PROCEDURE [XDCL, #GATE] mmp$init_system_privilege_map
  0  8730     (    offset: ost$segment_offset);
  0  8731
  0  8732     VAR
  0  8733       i: ost$segment,
  0  8734       leftover: boolean,
  0  8735       mapend: ost$segment,
  0  8736       mp_p: ^ost$system_privilege_map,
  0  8737       sdt_entry_p: ^mmt$segment_descriptor,
  0  8738       ste: ost$segment_descriptor,
  0  8739       xcb_p: ^ost$execution_control_block;
  0  8740
  0  8741 { Calculate where the map is in mainframe pageable.
  0  8742 { Note - hard-coded [1] must be changed if multiple task private segments exist.
  0  8743
  0  8744     mp_p := #ADDRESS (1, #SEGMENT (jmv$task_private_templ_p^.segment [1].content),
  4  8745           #OFFSET (jmv$task_private_templ_p^.segment [1].content) + offset);
  4  8746
  4  8747 { Find the system job XCB.
  4  8748
  4  8749     pmp$find_executing_task_xcb (xcb_p);
 34  8750
 34  8751 { Capture the system privilege segments.
 34  8752 { Insure no array bounds errors.
 34  8753
 34  8754     IF xcb_p^.xp.segment_table_length < UPPERBOUND (ost$system_privilege_map) THEN
 58  8755       leftover := TRUE;
 58  8756       mapend := xcb_p^.xp.segment_table_length;
 5E  8757     ELSE
 5E  8758       leftover := FALSE;
 5E  8759       mapend := UPPERBOUND (ost$system_privilege_map);
 66  8760     IFEND;
 66  8761
 66  8762 { Compute the bits for which both segment table entries and map entries exist.
 66  8763
 66  8764     FOR i := 0 TO mapend DO
 70  8765       sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, i);
 70  8766       ste := sdt_entry_p^.ste;
 70  8767       mp_p^ [i] := (ste.vl <> osc$vl_invalid_entry) AND (ste.xp <> osc$non_executable);
 B0  8768     FOREND;
 BA  8769
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$INIT_SYSTEM_PRIVILEGE_MAP

```
BA  8770 { Clear any leftover bits.
BA  8771
BA  8772     IF leftover THEN
BE  8773       FOR i := mapend + 1 TO UPPERBOUND (ost$system_privilege_map) DO
C8  8774         mp_p^ [i] := FALSE;
C8  8775       FOREND;
D4  8776     IFEND;
D4  8777
D4  8778   PROCEND mmp$init_system_privilege_map;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$ISSUE_RING1_SEGMENT_REQUEST

```
 0  8780
 0  8781   PROCEDURE [XDCL] mmp$issue_ring1_segment_request
 0  8782     (VAR rb: mmt$rb_ring1_segment_request);
 0  8783
 0  8784     VAR
 0  8785       count: integer,
 0  8786       status: ost$status,
 0  8787       sfid: gft$system_file_identifier;
 0  8788
 0  8789 {   Mmp$process_wmp_status (mtr) will set init_new_io to FALSE if the call is reissued for the wait option.
 0  8790
 0  8791     rb.init_new_io := TRUE;
 4  8792     FOR count := 1 TO 4 DO
16  8793       i#call_monitor (#LOC (rb), #SIZE (rb));
2E  8794       IF NOT rb.status.normal THEN
36  8795         IF rb.status.condition = mme$io_write_error THEN
46  8796           CASE rb.request OF
60  8797           = mmc$sr1_detach_file, mmc$sr1_flush_delete_seg_sfid, mmc$sr1_flush_seg_segnum =
60  8798             {Only attempt reallocate for these requests
60  8799             sfid := rb.sfid;
68  8800           ELSE
68  8801             RETURN;
6A  8802           CASEND;
6E  8803         ELSE
6E  8804           RETURN;
70  8805         IFEND;
74  8806       ELSE
74  8807         RETURN;
76  8808       IFEND;
76  8809       IF count = 4 THEN
7C  8810         RETURN;
7E  8811       IFEND;
7E  8812       dmp$reallocate_file_space (sfid, TRUE, status);
9A  8813       IF NOT status.normal THEN
A2  8814         RETURN;
A4  8815       IFEND;
A4  8816       rb.init_new_io := TRUE;
A4  8817     FOREND;
AC  8818
AC  8819   PROCEND mmp$issue_ring1_segment_request;
 0  8820
```

```
0  8822
0  8823 {
0  8824 {   The purpose of this procedure is to invalidate the specified segment
0  8825 { in the SDT table.  The backing store file is returned if one is assigned,
0  8826 { and the open_count in the FDE is zero.
0  8827 {
0  8828 {        MMP$INVALIDATE_SEGMENT (SEGMENT_NUMBER, VALIDATING_RING_NUMBER,
0  8829 {              SHARED_TASKID_ARRAY, STATUS)
0  8830 {
0  8831 { SEGMENT_NUMBER: (input) This parameter specifies the segment number to
0  8832 {        be invalidated.
0  8833 {
0  8834 { VALIDATING_RING_NUMBER: (input) This parameter specifies the validating
0  8835 {        ring_number of the request.
0  8836 {
0  8837 { SHARED_TASKID_ARRAY: (input) This parameter is a pointer to the list of
0  8838 {        taskids of tasks in which this segment is to be invalidated.
0  8839 {
0  8840 { STATUS: (output) This parameter is where the request status is returned
0  8841 {        to the caller.  The possible error codes are:
0  8842 {             dme$file_descriptor_not_deleted
0  8843 {             mme$invalid_close_segment_req
0  8844 {             mme$invalid_shared_taskid
0  8845 {             mme$segment_number_not_in_use
0  8846 {             mme$segment_number_too_big
0  8847 {
0  8848
0  8849   PROCEDURE [XDCL, #GATE] mmp$invalidate_segment
0  8850     (     segment_number: ost$segment;
0  8851           validating_ring_number: ost$valid_ring;
0  8852           shared_taskid_array: ^array [1 .. * ] of pmt$task_id;
0  8853      VAR status: ost$status);
0  8854
0  8855     VAR
0  8856       caller_id: ost$caller_identifier,
0  8857       fde_entry_p: gft$locked_file_desc_entry_p,
0  8858       i: integer,
0  8859       open_count: integer, {must be integer}
0  8860       pva: ^cell,
0  8861       rb: mmt$rb_ring1_segment_request,
0  8862       sdt_entry_p: ^mmt$segment_descriptor,
0  8863       sdtx_entry_p: ^mmt$segment_descriptor_extended,
0  8864       sdt_p: mmt$max_sdt_p,
0  8865       sdtx_p: mmt$max_sdtx_p,
0  8866       segnum: ost$segment,
0  8867       shadow_fde_p: gft$locked_file_desc_entry_p,
0  8868       shadow_open_count: gft$open_count,
0  8869       shadow_sfid: gft$system_file_identifier,
0  8870       task_xcb: ^ost$execution_control_block,
0  8871       task_sdt_entry_p: ^mmt$segment_descriptor,
0  8872       xcb_p: ^ost$execution_control_block;
0  8873
0  8874     IF shared_taskid_array <> NIL THEN
1E 8875       FOR i := LOWERBOUND (shared_taskid_array^) TO UPPERBOUND (shared_taskid_array^) DO
2E 8876         pmp$find_task_xcb (shared_taskid_array^ [i], task_xcb);
```



```
54  8877         IF task_xcb = NIL THEN
5E  8878           osp$set_status_abnormal ('MM', mme$invalid_shared_taskid, '', status);
94  8879           RETURN;
96  8880         IFEND;
96  8881       FOREND;
9C  8882     IFEND;
9C  8883
9C  8884     #CALLER_ID (caller_id);
9C  8885     mmp$fetch_sdt_sdtx_locked_fde (segment_number, sdt_entry_p, sdtx_entry_p, fde_entry_p, status);
D8  8886     IF status.normal = FALSE THEN
E0  8887       RETURN;
E2  8888     IFEND;
E2  8889
E2  8890     pva := #ADDRESS (caller_id.ring, segment_number, 0);
E2  8891
E2  8892 {***KLUDGE- the first conditional is necessary to delete task private in
E2  8893 {              job termination.
E2  8894     IF validating_ring_number > 2 THEN
102 8895       IF (sdtx_entry_p^.inheritance <> mmc$si_none) THEN
10E 8896         gfp$unlock_fde_p (fde_entry_p);
202 8897         osp$set_status_abnormal ('MM', mme$invalid_close_segment_req, '', status);
232 8898         RETURN;
234 8899       IFEND;
234 8900     IFEND; {***KLUDGE***}
234 8901
234 8902 {  Clear any segment locks left by the user.
234 8903
234 8904     IF sdtx_entry_p^.segment_lock <> mmc$lss_none THEN
240 8905       mmp$unlock_segment (pva, mmc$lus_free, osc$nowait, status);
262 8906       IF NOT status.normal THEN
26A 8907         gfp$unlock_fde_p (fde_entry_p);
35C 8908         osp$system_error ('Unexpected mmp$unlock_segment error', ^status);
380 8909       IFEND;
380 8910     IFEND;
380 8911
380 8912     pmp$find_executing_task_xcb (xcb_p);
386 8913     IF shared_taskid_array <> NIL THEN
3A6 8914       FOR i := LOWERBOUND (shared_taskid_array^) TO UPPERBOUND (shared_taskid_array^) DO
3B4 8915         pmp$find_task_xcb (shared_taskid_array^ [i], task_xcb);
3DA 8916         IF xcb_p <> task_xcb THEN
3EA 8917           task_sdt_entry_p := mmp$get_sdt_entry_p (task_xcb, segment_number);
3EA 8918           task_sdt_entry_p^.ste.v1 := osc$v1_invalid_entry;
3EA 8919           fde_entry_p^.open_count := fde_entry_p^.open_count - 1;
42A 8920         IFEND;
42A 8921       FOREND;
434 8922     IFEND;
434 8923
434 8924     open_count := 1;
434 8925     IF sdtx_entry_p^.shadow_info.shadow_segment_kind <> mmc$ssk_none THEN
442 8926       gfp$get_locked_fde_p (sdtx_entry_p^.shadow_info.shadow_sfid, shadow_fde_p);
570 8927       shadow_fde_p^.open_count := shadow_fde_p^.open_count - 1;
570 8928       shadow_open_count := shadow_fde_p^.open_count;
570 8929       gfp$unlock_fde_p (shadow_fde_p);
66E 8930       IF (shadow_open_count = 0) AND
686 8931           ((shadow_fde_p^.file_kind = gfc$fk_unnamed_file) OR
```

```
686  8932            (shadow_fde_p^.file_kind = gfc$fk_global_unnamed)) THEN
686  8933          destroy_segment (sdtx_entry_p^.shadow_info.shadow_sfid, shadow_fde_p,
6AC  8934              sfc$temp_file_space_limit, status);  {Can only destroy temp files}
6AC  8935        IFEND;
6B0  8936      ELSEIF sdtx_entry_p^.shadow_info.passive_for_shadow_by_segnum THEN
6B8  8937        pmp$find_executing_task_xcb (xcb_p);
6BE  8938        mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
6BE  8939        FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
714  8940          IF (sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND
748  8941                (sdtx_p^.sdtx_table [segnum].shadow_info.shadow_segment_number = segnum) AND
748  8942                (sdtx_p^.sdtx_table [segnum].shadow_info.shadow_segment_kind = mmc$ssk_segment_number) THEN
748  8943            sdtx_p^.sdtx_table [segnum].shadow_info.shadow_segment_kind := mmc$ssk_none;
748  8944            sdtx_p^.sdtx_table [segnum].shadow_info.passive_for_shadow_by_segnum := FALSE;
748  8945            open_count := open_count + 1;
752  8946          IFEND;
752  8947        FOREND;
756  8948      IFEND;
756  8949
756  8950      open_count  := fde_entry_p^.open_count - open_count;
756  8951      fde_entry_p^.open_count := open_count;
756  8952      gfp$unlock_fde_p (fde_entry_p);
85A  8953      IF open_count < 0 THEN
85E  8954        osp$system_error ('MM - neg open count in invalidate', NIL);
882  8955      IFEND;
882  8956
882  8957      IF (open_count = 0) AND ((fde_entry_p^.file_kind = gfc$fk_unnamed_file) OR
89A  8958            (fde_entry_p^.file_kind = gfc$fk_global_unnamed)) THEN
89A  8959        destroy_segment (sdtx_entry_p^.sfid, fde_entry_p, sdtx_entry_p^.file_limits_enforced, status);
8C0  8960      IFEND;
8C0  8961
8C0  8962      #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
8C2  8963      sdt_entry_p^.ste.v1 := osc$v1_invalid_entry;
8C2  8964
8C2  8965    PROCEND mmp$invalidate_segment;
```



```
 0  8967 {
 0  8968 {   The purpose of this request is to delete all segments unique to this
 0  8969 { job.  The ring 1 and 2 stack segments and the job fixed segment are
 0  8970 { not deleted.  These will be deleted in another phase of job termination.
 0  8971 {
 0  8972 { NOTE: This procedure can not be called from above ring 2.
 0  8973 {
 0  8974 {       MMP$JOB_DELETE_INHERITED_SDT (STATUS)
 0  8975 {
 0  8976 { STATUS: (output) This parameter is where the request status is returned.
 0  8977 {       No error codes are returned.  Osp$system_error is called if segments
 0  8978 {       cannot be deleted.
 0  8979 {
 0  8980
 0  8981
 0  8982   PROCEDURE [XDCL, #GATE] mmp$job_delete_inherited_sdt;
 0  8983
 0  8984     VAR
 0  8985       fde_entry_p: gft$locked_file_desc_entry_p,
 0  8986       rb: mmt$rb_ring1_segment_request,
 0  8987       ring_1_stack_segnum: ost$segment,
 0  8988       sdt_entry_p: ^mmt$segment_descriptor,
 0  8989       sdtx_entry_p: ^mmt$segment_descriptor_extended,
 0  8990       segnum: ost$segment,
 0  8991       status: ost$status,
 0  8992       xcb_p: ^ost$execution_control_block;
 0  8993
 0  8994     status.normal := TRUE;
 4  8995
 4  8996     pmp$find_executing_task_xcb (xcb_p);
10  8997     ring_1_stack_segnum := xcb_p^.xp.tos_registers [1].pva.seg;
10  8998
10  8999   /scan_sdt_for_inherited_segs/
10  9000     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
3C  9001       sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segnum);
3C  9002       IF (sdt_entry_p^.ste.v1 <> osc$v1_invalid_entry) THEN
6E  9003         sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
6E  9004         IF (sdtx_entry_p^.open_validating_ring_number > 0) AND (segnum <> osc$segnum_job_fixed_heap) AND
A6  9005               (segnum <> ring_1_stack_segnum) THEN
A6  9006           gfp$get_fde_p (sdtx_entry_p^.sfid, fde_entry_p);
FE  9007
FE  9008 {  Close segment (free memory and ASID and return backing store if it exists), have to call device
FE  9009 {  manager to return backing store during job termination.
FE  9010
FE  9011           IF (fde_entry_p^.media = gfc$fm_transient_segment) THEN
10A 9012             IF fde_entry_p^.asti <> 0 THEN
112  9013               rb.reqcode := syc$rc_ring1_segment_request;
112  9014               rb.request := mmc$sr1_delete_seg_segnum;
112  9015               rb.segnum := segnum;
112  9016               i#call_monitor (#LOC (rb), #SIZE (rb));
13E  9017             IFEND;
142  9018           ELSE
142  9019             fde_entry_p^.open_count := 0; {!!!### this is for fault tolerance - it should not be needed.}
142  9020             dmp$destroy_file (sdtx_entry_p^.sfid, sdtx_entry_p^.file_limits_enforced, status);
16A  9021             sdt_entry_p^.ste.v1 := osc$v1_invalid_entry; {Must be after call to destroy for job recovery!!}
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$JOB_DELETE_INHERITED_SDT

```
176  9022            IFEND;
176  9023
176  9024          IFEND;
176  9025        IFEND;
176  9026      FOREND /scan_sdt_for_inherited_segs/;
17A  9027
17A  9028      syp$return_jobs_r1_resources;
182  9029
182  9030    PROCEND mmp$job_delete_inherited_sdt;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$JOB_MULTIPROCESSING_CONTROL

```
 0  9032
 0  9033    PROCEDURE [XDCL] mmp$job_multiprocessing_control
 0  9034      (     enable: boolean;
 0  9035       VAR status: ost$status);
 0  9036
 0  9037      VAR
 0  9038        sdt_p: mmt$max_sdt_p,
 0  9039        sdtx_p: mmt$max_sdtx_p,
 0  9040        segnum: ost$segment,
 0  9041        xcb_p: ^ost$execution_control_block;
 0  9042
 0  9043      status.normal := TRUE;
 4  9044      pmp$find_executing_task_xcb (xcb_p);
14  9045
14  9046      mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
14  9047      FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
6A  9048        IF (sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND (segnum <> osc$segnum_job_fixed_heap) THEN
84  9049          IF (sdtx_p^.sdtx_table [segnum].inheritance = mmc$si_share_segment) AND
A2  9050               (sdtx_p^.sdtx_table [segnum].open_validating_ring_number <> 0) THEN
A2  9051            IF enable THEN
AA  9052              sdt_p^.st [segnum].ste.v1 := osc$v1_cache_bypass;
B8  9053            ELSE
B8  9054              sdt_p^.st [segnum].ste.v1 := osc$v1_regular_segment;
C2  9055            IFEND;
C2  9056          IFEND;
C2  9057        IFEND;
C2  9058      FOREND;
C6  9059
C6  9060    PROCEND mmp$job_multiprocessing_control;
 0  9061
```

```
      0  9063
      0  9064 {
      0  9065 {   The purpose of this procedure is to assign a device to a segment
      0  9066 { if memory manager needs it done.  This procedure is called by the ring 1
      0  9067 { trap handler when a specified system flag is set.
      0  9068 {
      0  9069 {         MMP$MFH_FOR_SEGMENT_MANAGER
      0  9070 {
      0  9071
      0  9072   PROCEDURE [XDCL, #GATE] mmp$mfh_for_segment_manager;
      0  9073
      0  9074     VAR
      0  9075       allocated_length: amt$file_byte_address,
      0  9076       ctime: 0 .. 0ffffffffffff(16),
      0  9077       gtid: ost$global_task_id,
      0  9078       status: ost$status,
      0  9079       xcb_p: ^ost$execution_control_block;
      0  9080
      0  9081 { Allow escaped allocation if the task has system tables locked.
      0  9082
      0  9083     pmp$find_executing_task_xcb (xcb_p);
      A  9084     IF xcb_p^.system_table_lock_count > 255 THEN
     2C  9085       xcb_p^.stlc_allocation := TRUE;
     2C  9086       RETURN;
     32  9087     IFEND;
     32  9088
     32  9089     mmp$process_file_alloc (allocated_length, status);
     4E  9090     IF NOT status.normal THEN
     56  9091       IF (status.condition <> dfe$family_not_served) AND (status.condition <> dfe$server_not_active) AND
     76  9092           (status.condition <> dfe$server_has_terminated) THEN
     76  9093         ctime := #FREE_RUNNING_CLOCK (0);
     7E  9094         REPEAT
     7E  9095           pmp$delay (1000, status);
     9A  9096           mmp$process_file_alloc (allocated_length, status);
     B2  9097         UNTIL (status.normal) OR ((#FREE_RUNNING_CLOCK (0) - ctime) > 10000000);
     CA  9098         IF NOT status.normal THEN
     D2  9099           pmp$get_executing_task_gtid (gtid);
     D8  9100           pmp$set_system_flag (mmc$failed_file_alloc_flag, gtid, status);
    116  9101           IF NOT status.normal THEN
    11E  9102             osp$system_error ('Error setting system flag-MMSMSC', NIL);
    142  9103           IFEND;
    142  9104         IFEND;
    142  9105       IFEND;
    142  9106     IFEND;
    142  9107
    142  9108   PROCEND mmp$mfh_for_segment_manager;
```



```
      0  9110
      0  9111   PROCEDURE [XDCL] mmp$mfh_shadow_file_reference;
      0  9112
      0  9113 { The purpose of this procedure is to move data from the shadowed file to the active
      0  9114 { file.  A page fault had occurred and the page was found to reside on the shadowed
      0  9115 { file.  Mmp$page_pull filled in the necessary information in the XCB and set the
      0  9116 { monitor flag, mmc$mf_shadow_file_reference.  The ring 1 trap handler called this
      0  9117 { procedure.
      0  9118
      0  9119     VAR
      0  9120       i: integer,
      0  9121       in_memory: boolean,
      0  9122       rma: integer,
      0  9123       status: ost$status,
      0  9124       xcb_p: ^ost$execution_control_block;
      0  9125
      0  9126     pmp$find_executing_task_xcb (xcb_p);
      A  9127     in_memory := TRUE;
      A  9128
      A  9129   /memory_check/
      A  9130     FOR i := 0 TO xcb_p^.shadow_reference_info.page_count - 1 DO
     30  9131       i#real_memory_address (#ADDRESS (1, #SEGMENT (xcb_p^.shadow_reference_info.source_pva),
     74  9132             #OFFSET (xcb_p^.shadow_reference_info.source_pva) + i * osv$page_size), rma);
     74  9133       IF rma < 0 THEN
     7C  9134         in_memory := FALSE;
     7C  9135         EXIT /memory_check/;
     84  9136       IFEND;
     84  9137     FOREND /memory_check/;
     88  9138
     88  9139     IF NOT in_memory THEN
     8C  9140       mmp$advise_in (xcb_p^.shadow_reference_info.source_pva,
     C2  9141             xcb_p^.shadow_reference_info.page_count * osv$page_size, status);
     C2  9142     IFEND;
     C2  9143     i#move (xcb_p^.shadow_reference_info.source_pva, xcb_p^.shadow_reference_info.destination_pva,
     C2  9144           xcb_p^.shadow_reference_info.page_count * osv$page_size);
    108  9145
    108  9146   PROCEND mmp$mfh_shadow_file_reference;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$MFH_VOLUME_UNAVAILABLE

```
      o  9148
      o  9149   PROCEDURE [XDCL] mmp$mfh_volume_unavailable;
      o  9150
      o  9151     VAR
      o  9152       gtid: ost$global_task_id,
      o  9153       mmv$pf_system_core,
      o  9154       mmv$pf_job_template: [XDCL] integer := 0,
      o  9155       psa: ^ost$minimum_save_area,
      o  9156       status: ost$status,
      o  9157       str1: integer,
      o  9158       str: string (80),
      o  9159       xcb: ^ost$execution_control_block;
      o  9160
      o  9161     psa := #PREVIOUS_SAVE_AREA ();
      4  9162     {This code assumes:
      4  9163     { Page fault for bad disk; trap to TH; TH calls this procedure
      4  9164     IF #RING (psa^.a2_previous_save_area) = 1 THEN
     16  9165
     16  9166       mmv$pf_system_core := mmv$pf_system_core + 1;
     16  9167       {We have interrupted the system core
     16  9168       {Allow rollback - then check for system tables locked
     16  9169       syp$cause_condition (syc$volume_unavailable);
     3C  9170
     3C  9171       {We still have control, so we must wait
     3C  9172       {If there are system resources tied up, we will be in trouble
     3C  9173       pmp$delay (30000, status);
     5A  9174
     5A  9175       {Return and attempt page fault again
     5A  9176     ELSE
     5A  9177
     5A  9178       mmv$pf_job_template := mmv$pf_job_template + 1;
     5A  9179       {We have interrupted the job template
     5A  9180       pmp$get_executing_task_gtid (gtid);
     6E  9181       pmp$set_system_flag (mmc$volume_unavailable_flag, gtid, status);
     AA  9182       IF NOT status.normal THEN
     B2  9183         osp$system_error ('Error setting system flag-MMSMSC', NIL);
     DA  9184       IFEND;
     DA  9185     IFEND;
     DA  9186
     DA  9187   PROCEND mmp$mfh_volume_unavailable;
      o  9188
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$MM_MOVE_MOD_SERVER_PAGE

```
      o  9190
      o  9191 {
      o  9192 { NAME:
      o  9193 {   MMP$MM_MOVE_MOD_SERVER_PAGE
      o  9194 {
      o  9195 { PURPOSE:
      o  9196 {   This procedure makes a request to move a single modified page from a source
      o  9197 { file to a destination file.  Only modified pages of the server file are moved;
      o  9198 { non-modified pages are discarded.  It is used to update the server image file
      o  9199 { on the client in the case of a server crash.  The procedure executes in job
      o  9200 { mode in a system task which writes the server image file with pages from all
      o  9201 { of the currently attached server files.
      o  9202 {
      o  9203 { ASSUMPTIONS:
      o  9204 {   The following assumptions are made in designing this interface:
      o  9205 {         . The state of the server will not change while this request is
      o  9206 {           removing pages
      o  9207 {         . The destination file (as specified by "destination_pva") is not on
      o  9208 {           the server
      o  9209 {         . All space for the destination file must be preallocated
      o  9210 {         . A page has not already been assigned to the destination pva
      o  9211 {         . Access to the server file whose page is being moved has been
      o  9212 {           inhibited by the SDTX access state
      o  9213 {         . The "destination_pva" is on a page boundry
      o  9214 {         . All server IO on the file has been dequeued and pages associated with
      o  9215 {           server IO have been unlocked
      o  9216 {         . This procedure is called only in the system job
      o  9217 {         . The File Descriptor for the file is expected to be locked and will
      o  9218 {           remain so during the monitor call
      o  9219 {
      o  9220 {          PROCEDURE [XDCL] mmp$mm_move_mod_server_page
      o  9221 {            (     sfid: gft$system_file_identifier;
      o  9222 {                  destination_pva: ^cell;
      o  9223 {              VAR byte_offset: ost$segment_offset;
      o  9224 {              VAR status: ost$status);
      o  9225 {
      o  9226 {   SFID: (INPUT) Specifies the System File ID of the file whose pages must be
      o  9227 {     moved to the destination file.
      o  9228 {   DESTINATION_PVA: (INPUT) Specifies the location within the server image file
      o  9229 {     to which a located modified page will be written.
      o  9230 {   BYTE_OFFSET: (OUTPUT) Specifies the beginning offset of the located page
      o  9231 {     which has been moved.
      o  9232 {   STATUS: (OUTPUT) Status processing.  Conditions which can be returned are:
      o  9233 {     mme$io_active_on_move_page      (retry)
      o  9234 {     mme$no_pages_found_for_move     (normal exit condition)
      o  9235 {     mme$page_table_full             (retry)
      o  9236 {
      o  9237
      o  9238   PROCEDURE [XDCL] mmp$mm_move_mod_server_page
      o  9239     (     system_file_id: gft$system_file_identifier;
      o  9240           destination_pva: ^cell;
      o  9241       VAR byte_offset: ost$segment_offset;
      o  9242       VAR status: ost$status);
      o  9243
      o  9244     VAR
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$MM_MOVE_MOD_SERVER_PAGE

```
      0  9245        rb_ring1_server_seg_request: mmt$rb_ring1_server_seg_request;
      0  9246
      0  9247      status.normal := TRUE;
      4  9248
      4  9249      rb_ring1_server_seg_request.reqcode := syc$rc_ring1_server_seg_request;
      4  9250      rb_ring1_server_seg_request.sfid := system_file_id;
      4  9251      rb_ring1_server_seg_request.request := mmc$ssr1_move_modified_df_page;
      4  9252      rb_ring1_server_seg_request.destination_pva := destination_pva;
      4  9253      rb_ring1_server_seg_request.byte_offset := 07fffffff(16); {dummy initialization}
      4  9254
      4  9255      i#call_monitor (#LOC (rb_ring1_server_seg_request), #SIZE (rb_ring1_server_seg_request));
     50  9256
     50  9257      byte_offset := rb_ring1_server_seg_request.byte_offset;
     50  9258      syp$set_status_from_mtr_status (rb_ring1_server_seg_request.status, status);
     9E  9259
     9E  9260    PROCEND mmp$mm_move_mod_server_page;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$OPEN_ASID_BASED_SEGMENT

```
      0  9262
      0  9263    PROCEDURE [XDCL] mmp$open_asid_based_segment
      0  9264      (     sdt_entry: mmt$segment_descriptor;
      0  9265            sdtx_entry: mmt$segment_descriptor_extended;
      0  9266       VAR segment_number: ost$segment;
      0  9267       VAR status: ost$status);
      0  9268
      0  9269      VAR
      0  9270        sdt_entry_p: ^mmt$segment_descriptor,
      0  9271        sdtx_entry_p: ^mmt$segment_descriptor_extended,
      0  9272        segnum: ost$segment,
      0  9273        xcb_p: ^ost$execution_control_block;
      0  9274
      0  9275      status.normal := TRUE;
      4  9276      pmp$find_executing_task_xcb (xcb_p);
     14  9277
     14  9278      find_available_segment_number (xcb_p, mmc$srs_not_reserved, segnum, status);
     40  9279      IF NOT status.normal THEN
     48  9280        RETURN;
     4A  9281      IFEND;
     4A  9282
     4A  9283 {  Add sdt_entry to the task's segment descriptor table (SDT)
     4A  9284      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segnum);
     4A  9285      sdt_entry_p^ := sdt_entry;
     4A  9286      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
     4A  9287      sdtx_entry_p^ := sdtx_entry;
     AE  9288
     AE  9289      segment_number := segnum;
     AE  9290
     AE  9291    PROCEND mmp$open_asid_based_segment;
      0  9292
```

```
      0  9294
      0  9295    PROCEDURE [XDCL, #GATE] mmp$open_file_by_sfid
      0  9296      (     sfid: gft$system_file_identifier;
      0  9297            r1: ost$valid_ring;
      0  9298            r2: ost$valid_ring;
      0  9299            sequential_random_selection: mmt$access_selections;
      0  9300            read_write_access_selection: mmt$segment_access_rights;
      0  9301       VAR segment_number: ost$segment;
      0  9302       VAR status: ost$status);
      0  9303
      0  9304      VAR
      0  9305        fde_entry_p: gft$locked_file_desc_entry_p,
      0  9306        page_streaming_ts_shift: 0 .. 15,
      0  9307        sdt_entry: mmt$segment_descriptor,
      0  9308        sdt_entry_p: ^mmt$segment_descriptor,
      0  9309        sdtx_entry_p: ^mmt$segment_descriptor_extended,
      0  9310        segment_res_state: mmt$segment_reservation_state,
      0  9311        segnum: ost$segment,
      0  9312        ste: mmt$segment_descriptor,
      0  9313        xcb_p: ^ost$execution_control_block;
      0  9314
      0  9315      status.normal := TRUE;
      4  9316      pmp$find_executing_task_xcb (xcb_p);
     14  9317
     14  9318 {  Find an available segment number if the caller did not supply one.
     14  9319
     14  9320      segment_res_state := mmc$srs_not_reserved;
     14  9321      find_available_segment_number (xcb_p, segment_res_state, segnum, status);
     40  9322      IF NOT status.normal THEN
     48  9323        RETURN;
     4A  9324      IFEND;
     4A  9325
     4A  9326 {  Add sdt_entry to the task's segment descriptor table (SDT) and the sdtx_entry to the
     4A  9327 {  segment descriptor table extended (SDTX).
     4A  9328
     4A  9329      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segnum);
     4A  9330      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
     4A  9331 { THIS SDTX ENTRY SHOULD BE THE DEV FILE
     4A  9332
     4A  9333      sdtx_entry_p^ := mmv$default_sdtx_entry;
     A4  9334      sdtx_entry_p^.sfid := sfid;
     A4  9335      sdt_entry := mmv$default_sdt_entry;
     A4  9336      sdt_entry.ste.r1 := r1;
     A4  9337      sdt_entry.ste.r2 := r2;
     A4  9338      gfp$get_locked_fde_p (sfid, fde_entry_p);
    1F4  9339      fde_entry_p^.open_count := fde_entry_p^.open_count + 1;
    1F4  9340      gfp$unlock_fde_p (fde_entry_p);
    2F4  9341      IF osv$multiple_cpus_possible THEN
    300  9342        sdt_entry.ste.v1 := osc$v1_cache_bypass;
    306  9343      IFEND;
    306  9344      IF read_write_access_selection = mmc$sar_read THEN
    30E  9345        sdt_entry.ste.wp := osc$non_writable;
    312  9346      IFEND;
    312  9347      IF sequential_random_selection = mmc$as_sequential THEN
    31A  9348        sdtx_entry_p^.software_attribute_set := $mmt$software_attribute_set
```



```
    31A  9349              [mmc$sa_read_transfer_unit, mmc$sa_free_behind];
    31A  9350        sdtx_entry_p^.stream.sequential_accesses := mmv$page_streaming_prestream;
    32C  9351      IFEND;
    32C  9352      mmp$convert_ps_transfer_size (16384, page_streaming_ts_shift); {force transfer size of 16384
    362  9353      sdtx_entry_p^.stream.transfer_size := page_streaming_ts_shift;
    362  9354      mmp$set_segment_access_rights (sdt_entry, sdtx_entry_p^);
    3A2  9355
    3A2  9356      store_ste_in_segment_table (sdt_entry, sfid, sdt_entry_p, fde_entry_p, segnum);
    3BC  9357
    3BC  9358      segment_number := segnum;
    3BC  9359
    3BC  9360    PROCEND mmp$open_file_by_sfid;
```

```
      0  9362 {
      0  9363 { The purpose of this procedure is to preallocate file space. The request will
      0  9364 {  attempt to allocate the number of bytes up to the length which is input by the user.
      0  9365 {
      0  9366 {       MMP$OS_PREALLOCATE_FILE_SPACE (PROCESS_VIRTUAL_ADDRESS, LENGTH, STATUS)
      0  9367 {
      0  9368 {  PROCESS_VIRTUAL_ADDRESS: (input) This parameter specifies the address of
      0  9369 {      the segment to which space is to be preallocated.
      0  9370 {
      0  9371 {  LENGTH: (input) This parameter specifies the desired length of the segment
      0  9372 {      after preallocation. This request will allocate (length - segment.eoi)
      0  9373 {      number of bytes.
      0  9374 {
      0  9375 {  STATUS: (output) This parameter will return the request status to the user.
      0  9376 {
      0  9377
      0  9378    PROCEDURE [XDCL, #GATE] mmp$os_preallocate_file_space
      0  9379      (     process_virtual_address: ^cell;
      0  9380            length: ost$segment_length;
      0  9381       VAR status: ost$status);
      0  9382
      0  9383      VAR
      0  9384        bytes_to_allocate: integer,
      0  9385        current_time: 0 .. 0fffffffffff(16),
      0  9386        delay_status: ost$status,
      0  9387        dfd_p: ^amt$disk_file_descriptor,
      0  9388        eoi: amt$file_byte_address,
      0  9389        segment_number: ost$segment,
      0  9390        sd_p: ^mmt$segment_descriptor,
      0  9391        sdtx_p: ^mmt$segment_descriptor_extended;
      0  9392
      0  9393      segment_number := #SEGMENT (process_virtual_address);
      4  9394      mmp$validate_segment_number (segment_number, sd_p, sdtx_p, status);
     36  9395      IF NOT status.normal THEN
     3E  9396        RETURN;
     40  9397      IFEND;
     40  9398
     40  9399      dmp$fetch_eoi (sdtx_p^.sfid, eoi, status);
     64  9400      IF NOT status.normal THEN
     6C  9401        RETURN;
     6E  9402      IFEND;
     6E  9403
     6E  9404      bytes_to_allocate := length - eoi;
     6E  9405      IF bytes_to_allocate <= 0 THEN
     7C  9406        RETURN;
     7E  9407      IFEND;
     7E  9408
     7E  9409      current_time := #FREE_RUNNING_CLOCK (0);
     86  9410      REPEAT
     86  9411        mmp$assign_mass_storage (segment_number, sdtx_p^.sfid, length, status);
     AE  9412        IF NOT status.normal AND (status.condition = dme$unable_to_alloc_all_space) THEN
     C6  9413          pmp$delay (1000, delay_status);
     E2  9414        IFEND;
     E2  9415      UNTIL (status.normal) OR ((#FREE_RUNNING_CLOCK (0) - current_time) > 60000000);
     FE  9416
```



```
     FE  9417    PROCEND mmp$os_preallocate_file_space;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$PRESET_PAGE_STREAMING

```
      0  9419
      0  9420   PROCEDURE [XDCL, #GATE] mmp$preset_page_streaming_r1
      0  9421     (     segment_number: ost$segment;
      0  9422           validating_ring_number: ost$valid_ring;
      0  9423           preset_and_save_fb_and_ts: boolean;
      0  9424           temp_transfer_size: integer;
      0  9425       VAR saved_transfer_size: 0 .. 15;
      0  9426       VAR saved_free_behind: boolean;
      0  9427       VAR status: ost$status);
      0  9428
      0  9429 {  Validate the pva and get a pointer to the segment descriptor.
      0  9430
      0  9431     VAR
      0  9432       page_streaming_ts_shift: 0 .. 15,
      0  9433       sdt_entry_p: ^mmt$segment_descriptor,
      0  9434       sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  9435
      0  9436     status.normal := TRUE;
      4  9437     mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
     36  9438     IF NOT status.normal THEN
     3E  9439       RETURN;
     40  9440     IFEND;
     40  9441
     40  9442 {  Verify that the pointer is in the read bracket of the segment.
     40  9443
     40  9444     IF (validating_ring_number > sdt_entry_p^.ste.r2) OR (validating_ring_number > sdt_entry_p^.ste.r2) THEN
     58  9445       osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
     8A  9446       RETURN;
     8C  9447     IFEND;
     8C  9448
     8C  9449     IF preset_and_save_fb_and_ts THEN
     94  9450
     94  9451 {  Save the current setting of transfer size and free behind.  Then set transfer size as specified,
     94  9452 {  and if the streaming boolean is false, set the preset_streaming boolean = TRUE.
     94  9453
     94  9454       saved_free_behind := (mmc$sa_free_behind IN sdtx_entry_p^.software_attribute_set);
     94  9455       IF NOT saved_free_behind THEN
     AC  9456         sdtx_entry_p^.software_attribute_set := sdtx_entry_p^.software_attribute_set +
     B4  9457               $mmt$software_attribute_set [mmc$sa_free_behind];
     B4  9458       IFEND;
     B4  9459       saved_transfer_size := sdtx_entry_p^.stream.transfer_size;
     B4  9460       mmp$convert_ps_transfer_size (temp_transfer_size, page_streaming_ts_shift);
    106  9461
    106  9462       IF sdtx_entry_p^.stream.transfer_size < page_streaming_ts_shift THEN
    11A  9463         sdtx_entry_p^.stream.transfer_size := page_streaming_ts_shift;
    122  9464       IFEND;
    122  9465       IF NOT sdtx_entry_p^.stream.streaming THEN
    130  9466         sdtx_entry_p^.stream.preset_streaming := TRUE;
    130  9467         IF sdtx_entry_p^.stream.sequential_accesses < mmv$page_streaming_prestream THEN
    148  9468           sdtx_entry_p^.stream.sequential_accesses := mmv$page_streaming_prestream;
    14C  9469         IFEND;
    14C  9470       IFEND;
    14E  9471
    14E  9472     ELSE
    14E  9473
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$PRESET_PAGE_STREAMING

```
    14E  9474 { reset SDTX with the saved transfer size and free behind from a previous call to mmp$preset_page_streaming
    14E  9475
    14E  9476       sdtx_entry_p^.stream.preset_streaming := FALSE;
    14E  9477       IF NOT saved_free_behind THEN
    168  9478         sdtx_entry_p^.software_attribute_set := sdtx_entry_p^.software_attribute_set *
    17A  9479               (-$mmt$software_attribute_set [mmc$sa_free_behind]);
    17A  9480       IFEND;
    17A  9481       IF sdtx_entry_p^.stream.transfer_size > saved_transfer_size THEN
    192  9482         sdtx_entry_p^.stream.transfer_size := saved_transfer_size;
    19A  9483       IFEND;
    19A  9484     IFEND;
    19A  9485
    19A  9486   PROCEND mmp$preset_page_streaming_r1;
      0  9487
```

```
      0  9489
      0  9490 {   The purpose of this procedure is to process file allocation
      0  9491 { if memory management has set the assign active flag in the SDTX.
      0  9492 { This procedure is either called as a result of a trap by the
      0  9493 { ring 1 trap handler or because a previous attempt to expand
      0  9494 { a file failed.
      0  9495 {   In the case of a previous failure, this procedure is called periodically
      0  9496 { from the ring 3 segment manager until it successfully expands the
      0  9497 { file, or the user terminates the task.
      0  9498 {
      0  9499 {    MMP$PROCESS_FILE_ALLOC (allocation_length, status)
      0  9500 {
      0  9501 { ALLOCATION_LENGTH: (output) This parameter returns the total amount of space
      0  9502 {       that was allocated by this request.
      0  9503 {
      0  9504 { STATUS: (output) This parameter returns the request status.
      0  9505 {
      0  9506
      0  9507   PROCEDURE [XDCL, #GATE] mmp$process_file_alloc
      0  9508     (VAR allocated_length: amt$file_byte_address;
      0  9509      VAR status: ost$status);
      0  9510
      0  9511     VAR
      0  9512       accumulated_allocated_length: amt$file_byte_address,
      0  9513       fde_p: gft$locked_file_desc_entry_p,
      0  9514       flush_pages: boolean,
      0  9515       rb: mmt$rb_ring1_segment_request,
      0  9516       segnum: integer,
      0  9517       sdt_p: mmt$max_sdt_p,
      0  9518       sdtx_p: mmt$max_sdtx_p,
      0  9519       tstatus: ost$status,
      0  9520       xcb_p: ^ost$execution_control_block;
      0  9521
      0  9522     status.normal := TRUE;
      4  9523     osp$begin_system_activity;
     16  9524     accumulated_allocated_length := 0;
     16  9525     rb.reqcode := syc$rc_ring1_segment_request;
     16  9526     rb.request := mmc$sr1_flush_avail_modified;
     16  9527     pmp$find_executing_task_xcb (xcb_p);
     2E  9528     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
     2E  9529
     2E  9530   /allocate_loop/
     2E  9531     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
     84  9532
     84  9533       IF segnum = 0 THEN
     88  9534         IF xcb_p^.assign_active_sfid <> gfv$null_sfid THEN
     A0  9535           gfp$get_locked_fde_p (xcb_p^.assign_active_sfid, fde_p);
    1CE  9536           IF fde_p <> NIL THEN
    1D8  9537             mmp$assign_mass_storage (0, xcb_p^.assign_active_sfid, 0, tstatus);
    200  9538             gfp$unlock_fde_p (fde_p);
    2F2  9539           IFEND;
    2F2  9540         IFEND;
    2F6  9541
    2F6  9542       ELSEIF (sdt_p^.st [segnum].ste.vl <> osc$vl_invalid_entry) AND
    324  9543             (sdtx_p^.sdtx_table [segnum].assign_active <> mmc$assign_active_null) THEN
```



```
    324  9544         rb.sfid := sdtx_p^.sdtx_table [segnum].sfid;
    324  9545         gfp$get_fde_p (rb.sfid, fde_p);
    380  9546         flush_pages := (fde_p^.media = gfc$fm_transient_segment);
    380  9547         mmp$assign_mass_storage (segnum, gfv$null_sfid, 0, tstatus);
    3B8  9548         IF tstatus.normal THEN
    3C0  9549           IF flush_pages THEN
    3C4  9550             i#call_monitor (#LOC (rb), #SIZE (rb));
    3E0  9551           IFEND;
    3E4  9552         ELSEIF (tstatus.condition = dme$unable_to_alloc_all_space) OR
    414  9553               (tstatus.condition = dme$unable_to_get_fd_lock) OR
    414  9554               (tstatus.condition = dfe$family_not_served) OR (tstatus.condition = dfe$server_not_active) OR
    414  9555               (tstatus.condition = dfe$server_has_terminated) THEN
    414  9556           IF (tstatus.condition = dme$unable_to_alloc_all_space) THEN
    420  9557             gfp$get_fde_p (rb.sfid, fde_p);
    476  9558             dmp$get_total_allocated_length (fde_p, allocated_length);
    492  9559             accumulated_allocated_length := accumulated_allocated_length + allocated_length;
    498  9560           IFEND;
    498  9561           status := tstatus;
    4B4  9562         ELSEIF tstatus.condition = dme$unable_to_create_fdt_entry THEN
    4BC  9563           syp$terminate_task (osc$rtr_sft_full);
    4CC  9564           EXIT /allocate_loop/;
    4D4  9565         ELSE
    4D4  9566           osp$end_system_activity;
    4DC  9567           syp$mfh_for_hang_task;
    4E4  9568         IFEND;
    4E4  9569       IFEND;
    4E4  9570     FOREND /allocate_loop/;
    4E8  9571
    4E8  9572     allocated_length := accumulated_allocated_length;
    4E8  9573
    4E8  9574     osp$end_system_activity;
    4F8  9575   PROCEND mmp$process_file_alloc;
      0  9576
```

```
      O  9578 [
      O  9579 {   The purpose of this request is to reserve a segment number within the
      O  9580 {   requesting task's
      O  9581 { address space for subsequent explicit assignment.  The reserved segment does
      O  9582 { not have any
      O  9583 { file or segment associated with it as a result of this request; the segment
      O  9584 { number is simply
      O  9585 { not chosen by memory management on any subsequent requests for an empty
      O  9586 { segment.
      O  9587 {   The segment can be used in subsequent requests that explicitly pass
      O  9588 {   segment management the
      O  9589 { segment number to be used.
      O  9590 {
      O  9591 {       MMP$RESERVE_SEGMENT_NUMBER (SEGMENT_NUMBER, STATUS)
      O  9592 {
      O  9593 { SEGMENT_NUMBER: (output) This parameter specifies the segment number that
      O  9594 { has been reserved.
      O  9595 {
      O  9596 { STATUS: (output) This parameter specifies the request status.
      O  9597 {
      O  9598
      O  9599
      O  9600   PROCEDURE [XDCL, #GATE] mmp$reserve_segment_number_r1
      O  9601     (     ada_stack_flag: boolean;
      O  9602      VAR segment_num_list: ^array [ * ] of ost$segment;
      O  9603      VAR status: ost$status);
      O  9604
      O  9605     VAR
      O  9606       segment_table_length: integer,
      O  9607       i: integer,
      O  9608       sdt_entry_p: ^mmt$segment_descriptor,
      O  9609       sdtx_entry_p: ^mmt$segment_descriptor_extended,
      O  9610       xcb_p: ^ost$execution_control_block,
      O  9611       segnum: ost$segment;
      O  9612
      O  9613     pmp$find_executing_task_xcb (xcb_p);
      A  9614
      A  9615     status.normal := TRUE;
      A  9616     segnum := mmv$first_transient_seg_index - 1;
      A  9617     segment_table_length := xcb_p^.xp.segment_table_length;
      A  9618
      A  9619     FOR i := LOWERBOUND (segment_num_list^) TO UPPERBOUND (segment_num_list^) DO
     58  9620       REPEAT
     58  9621         segnum := segnum + 1;
     58  9622         IF segnum > segment_table_length THEN
     5E  9623           expand_segment_table (xcb_p, status);
     70  9624           IF NOT status.normal THEN
     78  9625             RETURN;
     7A  9626           IFEND;
     7A  9627           segment_table_length := xcb_p^.xp.segment_table_length;
     86  9628         IFEND;
     86  9629         sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segnum);
     86  9630         sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
     86  9631       UNTIL (sdt_entry_p^.ste.vl = osc$vl_invalid_entry) AND
     E6  9632             (sdtx_entry_p^.segment_reservation_state = mmc$srs_not_reserved);
```



```
     E6  9633       IF ada_stack_flag THEN
     EE  9634         sdtx_entry_p^.segment_reservation_state := mmc$srs_reserved_shared_stack;
     F8  9635       ELSE
     F8  9636         sdtx_entry_p^.segment_reservation_state := mmc$srs_reserved;
     FE  9637       IFEND;
     FE  9638       segment_num_list^ [i] := segnum;
     FE  9639     FOREND;
    118  9640
    118  9641   PROCEND mmp$reserve_segment_number_r1;
```

```
      0  9643
      0  9644 {
      0  9645 {   The purpose of this procedure is to set the access mode of a segment
      0  9646 { based on its hardware attributes.
      0  9647 {
      0  9648 {         MMP$SET_ACCESS_MODE (SEGMENT_DESCRIPTOR, ACCESS_MODE)
      0  9649 {
      0  9650 { SEGMENT_DESCRIPTOR: (input) This parameter is the segment descriptor
      0  9651 {         from the segment table.
      0  9652 {
      0  9653 { ACCESS_MODE: (output) This parameter is where the access mode for the
      0  9654 {         specified segment descriptor is returned.
      0  9655 {
      0  9656
      0  9657   PROCEDURE [XDCL, #GATE] mmp$set_access_mode
      4  9658     (     segment_descriptor: ost$segment_descriptor;
      4  9659      VAR access_mode: pft$usage_selections);
      4  9660
      4  9661     access_mode := $pft$usage_selections [];
      4  9662
      4  9663     IF segment_descriptor.xp <> osc$non_executable THEN
     1A  9664       access_mode := access_mode + $pft$usage_selections [pfc$execute];
     22  9665     IFEND;
     22  9666
     22  9667     IF segment_descriptor.rp <> osc$non_readable THEN
     2E  9668       access_mode := access_mode + $pft$usage_selections [pfc$read];
     3C  9669     IFEND;
     3C  9670
     3C  9671     IF segment_descriptor.wp <> osc$non_writable THEN
     48  9672       access_mode := access_mode + $pft$usage_selections [pfc$shorten] + $pft$usage_selections [pfc$append] +
     5C  9673             $pft$usage_selections [pfc$modify];
     5C  9674     IFEND;
     5C  9675
     5C  9676   PROCEND mmp$set_access_mode;
```



```
      0  9678
      0  9679   PROCEDURE [XDCL, #GATE] mmp$set_access_selections_r1
      0  9680     (     segment_number: ost$segment;
      0  9681           validating_ring_number: ost$valid_ring;
      0  9682           access_selections: mmt$access_selections;
      0  9683      VAR status: ost$status);
      0  9684
      0  9685     VAR
      0  9686       sd_p: ^mmt$segment_descriptor,
      0  9687       sdtx_p: ^mmt$segment_descriptor_extended;
      0  9688
      0  9689 {  Validate the pva and get a pointer to the segment descriptor.
      0  9690
      0  9691     mmp$validate_segment_number (segment_number, sd_p, sdtx_p, status);
     30  9692     IF NOT status.normal THEN
     38  9693       RETURN;
     3A  9694     IFEND;
     3A  9695
     3A  9696 {  Verify that the pointer is in the read bracket of the segment.
     3A  9697
     3A  9698     IF (validating_ring_number > sd_p^.ste.r2) THEN
     4E  9699       osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
     80  9700       RETURN;
     82  9701     IFEND;
     82  9702     IF access_selections = mmc$as_sequential THEN
     8C  9703       sdtx_p^.software_attribute_set := sdtx_p^.software_attribute_set +
     A0  9704             $mmt$software_attribute_set [mmc$sa_read_transfer_unit, mmc$sa_free_behind];
     A0  9705     ELSEIF access_selections = mmc$as_random THEN
     A6  9706       sdtx_p^.software_attribute_set := sdtx_p^.software_attribute_set -
     BA  9707             $mmt$software_attribute_set [mmc$sa_read_transfer_unit, mmc$sa_free_behind];
     BA  9708     ELSEIF access_selections = mmc$as_read_tu THEN
     C0  9709       sdtx_p^.software_attribute_set := sdtx_p^.software_attribute_set +
     D4  9710             $mmt$software_attribute_set [mmc$sa_read_transfer_unit] -
     D4  9711             $mmt$software_attribute_set [mmc$sa_free_behind];
     D4  9712     IFEND;
     D4  9713
     D4  9714     IF (mmc$sa_read_transfer_unit IN sdtx_p^.software_attribute_set) THEN
     E4  9715       IF sdtx_p^.stream.sequential_accesses < mmv$page_streaming_prestream THEN
     F4  9716         sdtx_p^.stream.sequential_accesses := mmv$page_streaming_prestream;
     F8  9717       IFEND;
     F8  9718     IFEND;
     F8  9719   PROCEND mmp$set_access_selections_r1;
      0  9720
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$SET_SEGMENT_ACCESS_RIGHTS

```
      0  9722
      0  9723   PROCEDURE [XDCL, INLINE] mmp$set_segment_access_rights
      4  9724     (     sd: mmt$segment_descriptor;
      4  9725      VAR sdtx: mmt$segment_descriptor_extended);
      4  9726
      4  9727     IF sd.ste.wp = osc$non_writable THEN
     10  9728       sdtx.access_rights := mmc$sar_read;
     1C  9729     ELSEIF mmc$sa_no_append IN sdtx.software_attribute_set THEN
     2C  9730       sdtx.access_rights := mmc$sar_modify;
     34  9731     ELSE
     34  9732       sdtx.access_rights := mmc$sar_write_extend;
     3A  9733     IFEND;
     3A  9734
     3A  9735   PROCEND mmp$set_segment_access_rights;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$SET_SEGMENT_LENGTH_R1

```
      0  9737
      0  9738   PROCEDURE [XDCL, #GATE] mmp$set_segment_length_r1
      0  9739     (     segment_number: ost$segment;
      0  9740           validating_ring_number: ost$valid_ring;
      0  9741           segment_length: ost$segment_length;
      0  9742      VAR status: ost$status);
      0  9743
      0  9744 {
      0  9745 {   The purpose of this procedure is to set or get the segment length for the
      0  9746 { specified segment.  Whether to set or get segment length is based on the
      0  9747 { 'set_or_get_segment_length' parameter.
      0  9748 {
      0  9749
      0  9750     VAR
      0  9751       request_block: mmt$rb_set_get_segment_length,
      0  9752       sdt_entry_p: ^mmt$segment_descriptor,
      0  9753       sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  9754
      0  9755     mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
     30  9756     IF status.normal = FALSE THEN
     38  9757       RETURN;
     3A  9758     IFEND;
     3A  9759
     3A  9760 {  Verify that caller is within write bracket and has write access.
     3A  9761
     3A  9762     IF validating_ring_number > sdt_entry_p^.ste.r1 THEN
     4E  9763       osp$set_status_abnormal ('MM', mme$caller_not_in_write_bracket, '', status);
     80  9764       RETURN;
     86  9765     ELSEIF sdt_entry_p^.ste.wp = osc$non_writable THEN
     92  9766       osp$set_status_abnormal ('MM', mme$no_write_access, '', status);
     C4  9767       RETURN;
     C8  9768     IFEND;
     C8  9769
     C8  9770     gfp$get_fde_p (sdtx_entry_p^.sfid, request_block.fde_p);
    128  9771     request_block.request_code := syc$rc_set_get_segment_length;
    128  9772     request_block.subfunction_code := mmc$sf_set_segment_length_fde_p;
    128  9773     request_block.segment_length := segment_length;
    128  9774
    128  9775     i#call_monitor (#LOC (request_block), #SIZE (request_block));
    156  9776
    156  9777   PROCEND mmp$set_segment_length_r1;
      0  9778
```

```
      0  9780
      0  9781    PROCEDURE [XDCL, #GATE] mmp$store_segment_attributes_r1
      0  9782      (     segment_number: ost$segment;
      0  9783            validating_ring: ost$valid_ring;
      0  9784            segment_attributes: ^array [ * ] of mmt$attribute_descriptor;
      0  9785       VAR status: ost$status);
      0  9786
      0  9787      VAR
      0  9788        access_mode: pft$usage_selections,
      0  9789        scratch_segment_number: ost$segment,
      0  9790        fde_entry_p: gft$locked_file_desc_entry_p,
      0  9791        i: integer,
      0  9792        pva: ^cell,
      0  9793        sdt_entry_p: ^mmt$segment_descriptor,
      0  9794        sdtx_entry_p: ^mmt$segment_descriptor_extended;
      0  9795
      0  9796 {  Validate the pva and get a pointer to the segment descriptor.
      0  9797
      0  9798      mmp$validate_segment_number (segment_number, sdt_entry_p, sdtx_entry_p, status);
     30  9799      IF NOT status.normal THEN
     38  9800        RETURN;
     3A  9801      IFEND;
     3A  9802
     3A  9803      IF (validating_ring > sdtx_entry_p^.open_validating_ring_number) THEN
     4A  9804        osp$set_status_abnormal ('MM', mme$ring_violation, '', status);
     7C  9805        RETURN;
     7E  9806      IFEND;
     7E  9807
     7E  9808 {  Validate that attributes can be modified.
     7E  9809
     7E  9810      FOR i := LOWERBOUND (segment_attributes^) TO UPPERBOUND (segment_attributes^) DO
     9E  9811        CASE segment_attributes^ [i].keyword OF
     E2  9812        = mmc$kw_null_keyword =
     E6  9813        = mmc$kw_ring_numbers =
     E6  9814          sdt_entry_p^.ste.r1 := segment_attributes^ [i].r1;
     E6  9815          sdt_entry_p^.ste.r2 := segment_attributes^ [i].r2;
    10A  9816        = mmc$kw_max_segment_length =
    10A  9817          gfp$get_locked_fde_p (sdtx_entry_p^.sfid, fde_entry_p);
    240  9818          fde_entry_p^.file_limit := segment_attributes^ [i].max_length;
    240  9819          gfp$unlock_fde_p (fde_entry_p);
    356  9820        = mmc$kw_error_exit_procedure =
    356  9821          osp$set_status_abnormal ('MM', mme$unsupported_keyword, '', status);
    388  9822          RETURN;
    38E  9823        = mmc$kw_hardware_attributes =
    38E  9824          IF (mmc$ha_execute_local IN segment_attributes^ [i].hardware_attri_set) AND (validating_ring > 3) THEN
    3B6  9825            osp$set_status_abnormal ('MM', mme$execute_local_invalid, '', status);
    3E8  9826            RETURN;
    3EA  9827          IFEND;
    3EA  9828
    3EA  9829          IF mmc$ha_read IN segment_attributes^ [i].hardware_attri_set THEN
    408  9830            sdt_entry_p^.ste.rp := osc$read_uncontrolled;
    41E  9831          ELSE
    41E  9832            sdt_entry_p^.ste.rp := osc$non_readable;
    432  9833          IFEND;
    432  9834          IF mmc$ha_binding IN segment_attributes^ [i].hardware_attri_set THEN
```



```
    450  9835            IF validating_ring <= 3 THEN
    45A  9836              sdt_entry_p^.ste.rp := osc$binding_segment;
    46E  9837            ELSE
    46E  9838              osp$set_status_abnormal ('MM', mme$binding_attribute_invalid, '', status);
    4A0  9839              RETURN;
    4A2  9840            IFEND;
    4A2  9841          IFEND;
    4A2  9842          IF mmc$ha_write IN segment_attributes^ [i].hardware_attri_set THEN
    4C0  9843            sdt_entry_p^.ste.wp := osc$write_uncontrolled;
    4D4  9844          ELSE
    4D4  9845            sdt_entry_p^.ste.wp := osc$non_writable;
    4E6  9846          IFEND;
    4E6  9847          IF mmc$ha_cache_bypass IN segment_attributes^ [i].hardware_attri_set THEN
    504  9848            sdt_entry_p^.ste.vl := osc$vl_cache_bypass;
    51A  9849          ELSE
    51A  9850            sdt_entry_p^.ste.vl := osc$vl_regular_segment;
    52A  9851          IFEND;
    52A  9852          IF mmc$ha_execute IN segment_attributes^ [i].hardware_attri_set THEN
    548  9853            sdt_entry_p^.ste.xp := osc$non_privileged;
    55C  9854          ELSE
    55C  9855            IF mmc$ha_execute_local IN segment_attributes^ [i].hardware_attri_set THEN
    564  9856              sdt_entry_p^.ste.xp := osc$local_privilege;
    578  9857            ELSE
    578  9858              IF (mmc$ha_execute_global IN segment_attributes^ [i].hardware_attri_set) AND
    588  9859                  (validating_ring > 1) THEN
    588  9860                osp$set_status_abnormal ('MM', mme$execute_global_invalid, '', status);
    5BA  9861                RETURN;
    5C0  9862              ELSE
    5C0  9863                sdt_entry_p^.ste.xp := osc$non_executable;
    5D2  9864              IFEND;
    5D2  9865            IFEND;
    5D2  9866          IFEND;
    5D6  9867        = mmc$kw_segment_access_control =
    5D6  9868          IF segment_attributes^ [i].access_control.execute_privilege = osc$local_privilege THEN
    5F2  9869            osp$set_status_abnormal ('MM', mme$execute_local_invalid, '', status);
    624  9870          IFEND;
    624  9871
    624  9872          IF segment_attributes^ [i].access_control.cache_bypass = TRUE THEN
    63E  9873            sdt_entry_p^.ste.vl := osc$vl_cache_bypass;
    654  9874          ELSE
    654  9875            sdt_entry_p^.ste.vl := osc$vl_regular_segment;
    664  9876          IFEND;
    664  9877
    664  9878          IF (segment_attributes^ [i].access_control.execute_privilege = osc$global_privilege) AND
    688  9879              (validating_ring > 1) THEN
    688  9880            osp$set_status_abnormal ('MM', mme$execute_global_invalid, '', status);
    6BA  9881            RETURN;
    6C0  9882          ELSE
    6C0  9883            sdt_entry_p^.ste.xp := segment_attributes^ [i].access_control.execute_privilege;
    6E8  9884          IFEND;
    6E8  9885
    6E8  9886          IF (segment_attributes^ [i].access_control.read_privilege = osc$binding_segment) AND
    70C  9887              (validating_ring > 3) THEN
    70C  9888            osp$set_status_abnormal ('MM', mme$binding_attribute_invalid, '', status);
    73C  9889            RETURN;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$STORE_SEGMENT_ATTRIBUTES_R1

```
     73E  9890          IFEND;
     73E  9891          sdt_entry_p^.ste.rp := segment_attributes^ [i].access_control.read_privilege;
     73E  9892          sdt_entry_p^.ste.wp := segment_attributes^ [i].access_control.write_privilege;
     776  9893        ELSE
     776  9894          osp$set_status_abnormal ('MM', mme$set_unmodifiable_attribute, '', status);
     7A8  9895          RETURN;
     7AA  9896        CASEND;
     7AA  9897      FOREND;
     7AE  9898
     7AE  9899      pva := #ADDRESS (1, segment_number, 0);
     7AE  9900      #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
     7C4  9901
     7C4  9902    PROCEND mmp$store_segment_attributes_r1;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$TASK_DELETE_INHERITED_SDT

```
       0  9904 {
       0  9905 {   The purpose of this request is to delete all segments with the
       0  9906 { inheritance attribute of mmc$si_new_segment. These segments are primarily
       0  9907 { the task template segments.  The SDT and SDTX tables are deallocated.
       0  9908 { This request is used during task termination.
       0  9909 {
       0  9910 {       MMP$TASK_DELETE_INHERITED_SDT (TASK_ID, STATUS)
       0  9911 {
       0  9912 { TASK_ID: (input) This parameter identifies the task being terminated.
       0  9913 {
       0  9914 { STATUS: (output) This parameter is where the request status is returned.
       0  9915 {         The possible error codes are:
       0  9916 {               mme$invalid_pva
       0  9917 {               mme$invalid_task_id
       0  9918 {               mme$ref_to_unrecovered_file
       0  9919
       0  9920   PROCEDURE [XDCL, #GATE] mmp$task_delete_inherited_sdt
       0  9921     (     task_id: pmt$task_id;
       0  9922      VAR status: ost$status);
       0  9923
       0  9924     VAR
       0  9925       fde_entry_p: gft$file_desc_entry_p,
       0  9926       local_sdt_p: ^cell,
       0  9927       local_sdtx_p: ^cell,
       0  9928       rb: mmt$rb_ring1_segment_request,
       0  9929       sdt_p: mmt$max_sdt_p,
       0  9930       sdtx_p: mmt$max_sdtx_p,
       0  9931       segnum: ost$segment,
       0  9932       xcb_p: ^ost$execution_control_block;
       0  9933
       0  9934     status.normal := TRUE;
       4  9935     pmp$find_task_xcb (task_id, xcb_p);
      2A  9936     IF xcb_p = NIL THEN
      38  9937       osp$set_status_abnormal ('MM', mme$invalid_task_id, '', status);
      6A  9938       RETURN;
      6C  9939     IFEND;
      6C  9940
      6C  9941     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
      6C  9942
      6C  9943   /scan_sdt_for_inherited_segs/
      6C  9944     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
      B4  9945       IF (sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND
      E0  9946             (sdtx_p^.sdtx_table [segnum].inheritance = mmc$si_new_segment) THEN
      E0  9947
      E0  9948 {  Close or delete the segment based on whether it is assigned to a file or not.
      E0  9949         gfp$get_fde_p (sdtx_p^.sdtx_table [segnum].sfid, fde_entry_p);
     134  9950         IF fde_entry_p^.open_count <> 1 THEN
     140  9951           osp$system_error (' FDE.OPEN_COUNT incorrect', NIL);
     164  9952         IFEND;
     164  9953         fde_entry_p^.open_count := fde_entry_p^.open_count - 1;
     164  9954         destroy_segment (sdtx_p^.sdtx_table [segnum].sfid, fde_entry_p,
     1A0  9955               sdtx_p^.sdtx_table [segnum].file_limits_enforced, status);
     1A0  9956       IFEND;
     1A0  9957     FOREND /scan_sdt_for_inherited_segs/;
     1A4  9958
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$TASK_DELETE_INHERITED_SDT

```
1A4  9959 {  Release the SDT and SDTX table space.
1A4  9960
1A4  9961      local_sdt_p := #ADDRESS (1, osc$segnum_job_fixed_heap, xcb_p^.sdt_offset);
1A4  9962      local_sdtx_p := #ADDRESS (1, osc$segnum_job_fixed_heap, xcb_p^.sdtx_offset);
1A4  9963      IF #SIZE (local_sdt_p^) > osv$page_size THEN
1D6  9964        mmp$free_pages (#ADDRESS (1, #SEGMENT (local_sdt_p), #OFFSET (local_sdt_p)), #SIZE (local_sdt_p^),
208  9965             osc$wait, status);
208  9966      IFEND;
208  9967      FREE local_sdtx_p IN osv$job_fixed_heap^;
22E  9968      FREE local_sdt_p IN osv$job_fixed_heap^;
250  9969
250  9970    PROCEND mmp$task_delete_inherited_sdt;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$TERMINATE_SHADOWING_R1

```
 0  9972 {
 0  9973 {   The purpose of this request is to invalidate an ACTIVE segment in the SDT.
 0  9974 {   The backing store file is destroyed and returned if one is assigned.
 0  9975 {
 0  9976 {       MMP$TERMINATE_SHADOWING_R1 (SEGMENT_NUMBER, STATUS);
 0  9977 {
 0  9978 { SEGMENT_NUMBER: (input) This parameter specifies the SEGMENT NUMBER
 0  9979 {       to be invalidated.
 0  9980 {
 0  9981 { STATUS: (output) This parameter specifies the request status returned.
 0  9982 {
 0  9983
 0  9984
 0  9985   PROCEDURE [XDCL, #GATE] mmp$terminate_shadowing_r1
 0  9986     (    segment_number: ost$segment;
 0  9987      VAR status: ost$status);
 0  9988
 0  9989     VAR
 0  9990       fde_entry_p: gft$locked_file_desc_entry_p,
 0  9991       open_count: gft$open_count,
 0  9992       pva: ^cell,
 0  9993       rb: mmt$rb_ring1_segment_request,
 0  9994       sdt_entry_p: ^mmt$segment_descriptor,
 0  9995       sdtx_entry_p: ^mmt$segment_descriptor_extended;
 0  9996
 0  9997
 0  9998     pva := #ADDRESS (1, segment_number, 0);
 4  9999     mmp$fetch_sdt_sdtx_locked_fde (segment_number, sdt_entry_p, sdtx_entry_p, fde_entry_p, status);
48 10000     IF status.normal = FALSE THEN
50 10001       RETURN;
52 10002     IFEND;
52 10003
52 10004
52 10005 {  Verify if shadow active. Only READ_WRITE shadowed segments can be terminated
52 10006
52 10007     IF sdtx_entry_p^.shadow_info.shadow_segment_kind <> mmc$ssk_read_write_file THEN
5E 10008       osp$set_status_abnormal ('MM', mme$invalid_shadow_segment, '', status);
8E 10009       gfp$unlock_fde_p (fde_entry_p);
17A 10010      RETURN;
17C 10011    IFEND;
17C 10012
17C 10013
17C 10014 {  Clear any segment locks left by the user.
17C 10015
17C 10016    IF sdtx_entry_p^.segment_lock <> mmc$lss_none THEN
188 10017      mmp$unlock_segment (pva, mmc$lus_free, osc$nowait, status);
1A8 10018      IF NOT status.normal THEN
1B0 10019        gfp$unlock_fde_p (fde_entry_p);
2A6 10020        osp$system_error ('Unexpected mmp$unlock_segment error', ^status);
2CA 10021      IFEND;
2CA 10022    IFEND;
2CA 10023
2CA 10024 { Decrement active file open count. Delete the active FDE if open count is now zero.
2CA 10025
2CA 10026    fde_entry_p^.open_count := fde_entry_p^.open_count - 1;
```

NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$TERMINATE_SHADOWING_R1

```
2CA 10027      open_count := fde_entry_p^.open_count;
2CA 10028      gfp$unlock_fde_p (fde_entry_p);
3CC 10029
3CC 10030      IF open_count = 0 THEN
3D0 10031        destroy_segment (sdtx_entry_p^.sfid, fde_entry_p, sdtx_entry_p^.file_limits_enforced, status);
3EE 10032      IFEND;
3EE 10033
3EE 10034 {  Change segment table entry to be unshadowed.
3EE 10035
3EE 10036      sdt_entry_p^.ste.asid := 0;
3EE 10037      sdtx_entry_p^.sfid := sdtx_entry_p^.shadow_info.shadow_sfid;
3EE 10038      sdtx_entry_p^.shadow_info.shadow_segment_kind := mmc$ssk_none;
3EE 10039      sdtx_entry_p^.shadow_info.passive_for_shadow_by_segnum := FALSE;
3EE 10040
3EE 10041      #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
40C 10042
40C 10043    PROCEND mmp$terminate_shadowing_r1;
```


NOS/VE - MMM$SEGMENT_MANAGER_SYSTEM_CORE
MMP$VALIDATE_SEGMENT_NUMBER

```
  0 10045
  0 10046 {
  0 10047 {    The purpose of this procedure is to validate a segment.  Pointers to the
  0 10048 { segment's SDT and SDTX entries are returned.  The segment number is
  0 10049 { checked to see if it is within the bounds of the segment table and whether the
  0 10050 { valid bit is set.
  0 10051 {
  0 10052 {         MMP$VALIDATE_SEGMENT_NUMBER (SEGMENT_NUMBER, SDT_ENTRY_P,
  0 10053 {           SDTX_ENTRY_P, STATUS)
  0 10054 {
  0 10055 { SEGMENT_NUMBER: (input) This parameter specifies the segment to be
  0 10056 {         validated.
  0 10057 {
  0 10058 { SDT_ENTRY_P: (output) This parameter is where a pointer to the SDT entry
  0 10059 {         is returned.
  0 10060 {
  0 10061 { SDTX_ENTRY_P: (output) This parameter is where a pointer to the SDTX entry
  0 10062 {         is returned.
  0 10063 {
  0 10064 { STATUS: (output) This parameter is where the request status is returned
  0 10065 {         to the caller.  The possible error codes are:
  0 10066 {               mme$segment_number_not_in_use
  0 10067 {               mme$segment_number_too_big
  0 10068 {
  0 10069
  0 10070
  0 10071    PROCEDURE [XDCL, #GATE] mmp$validate_segment_number
  0 10072      (     segment_number: ost$segment;
  0 10073       VAR sdt_entry_p: ^mmt$segment_descriptor;
  0 10074       VAR sdtx_entry_p: ^mmt$segment_descriptor_extended;
  0 10075       VAR status: ost$status);
  0 10076
  0 10077      VAR
  0 10078        xcb_p: ^ost$execution_control_block;
  0 10079
  0 10080      status.normal := TRUE;
  4 10081      pmp$find_executing_task_xcb (xcb_p);
 14 10082
 14 10083      IF segment_number > xcb_p^.xp.segment_table_length THEN
 38 10084        IF segment_number > 4095 THEN
 40 10085          osp$set_status_abnormal ('MM', mme$segment_number_too_big, '', status);
 74 10086        ELSE
 74 10087          osp$set_status_abnormal ('MM', mme$segment_number_not_in_use, '', status);
 A6 10088        IFEND;
 A6 10089        RETURN;
 A8 10090      IFEND;
 A8 10091
 A8 10092      sdt_entry_p := mmp$get_sdt_entry_p (xcb_p, segment_number);
 A8 10093      sdtx_entry_p := mmp$get_sdtx_entry_p (xcb_p, segment_number);
 A8 10094      IF sdt_entry_p^.ste.v1 = osc$v1_invalid_entry THEN
116 10095        osp$set_status_abnormal ('MM', mme$segment_number_not_in_use, '', status);
148 10096      IFEND;
148 10097
148 10098    PROCEND mmp$validate_segment_number;
  0 10100 MODEND mmm$segment_manager_system_core;
```

**** 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    7510   Code scheduling abandoned for this block due to register jamming.
WARNING  CY 821    7544   Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY
   ****   2    warning diagnostics
```

IDENTIFIER	DEFINED ON LINE	REFERENCES							
a2_previous_save_area	4877	9164							
access_control	3860	7805	7811	7816	7819	7824	7825	8508/M	8510/M
		8512/M	8513/M	8514/M	9868	9872	9878	9883	9886
		9891	9892						
access_mode	9659	9661/M	9664/M	9664	9668/M	9668	9672/M	9672	
access_rights	4495	7507/M	7507/M	7507/M	7543/M	7543/M	7543/M	8691/M	9354/M
		9354/M	9354/M	9728/M	9730/M	9732/M			
access_selections	9682	9702	9705	9708					
accumulated_allocated_length	9512	9524/M	9559/M	9559	9572				
ada_stack_flag	9601	9633							
add_sdt_sdtx_entry	7447	7555	7936	8018					
addr	5458	8560/M							
addr_list	8530	8560/S	8560/M	8561/S	8561/M	8566/P			
addr_returned	8531	8558/M	8566/P						
allocate_loop	9530	9530	9564	9570					
allocated_length	8536	8554/P	8556						
allocated_length	9075	9089/P	9096/P						
allocated_length	9508	9558/P	9559	9572/M					
allocation_unit_size	2009	8685/M							
already_locked	5572	5577	5577						
already_locked	5588	5612	5613						
already_locked	5789	5797	5797						
already_locked	7447	7514	7514						
already_locked	7572	7612	7612						
already_locked	7662	7712	7712	7717	7717				
already_locked	8121	8243	8243						
already_locked	8390	8417	8417						
already_locked	8526	8552	8552						
already_locked	8849	8926	8926						
already_locked	9295	9338	9338						
already_locked	9507	9535	9535						
already_locked	9781	9817	9817						
amc$access_mode	1134	611							
amc$average_record_length	1136	686							
amc$block_type	1137	613							
amc$cell_pointer	16	20							
amc$character_conversion	1138	615							
amc$clear_space	1139	617							
amc$collate_table_name	1141	688							
amc$compression_procedure_name	1194	690							
amc$data_padding	1142	693							
amc$dynamic_home_block_space	1195	695							
amc$embedded_key	1143	697							
amc$error_exit_name	1144	619							
amc$error_limit	1146	699							
amc$error_options	1147	621							
amc$estimated_record_count	1148	701							
amc$file_access_procedure	1149	623							
amc$file_byte_limit	9	12	438	895	933	969	1248	1310	
amc$file_contents	1150	625							
amc$file_limit	1152	627							
amc$file_organization	1153	629							
amc$file_processor	1154	631							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
amc$file_structure	1155	633							
amc$forced_write	1156	635							
amc$hashing_procedure_name	1196	703							
amc$heap_pointer	16	22							
amc$index_levels	1162	705							
amc$index_padding	1163	707							
amc$initial_home_block_count	1197	709							
amc$internal_code	1164	637							
amc$key_length	1165	711							
amc$key_position	1166	713							
amc$key_type	1167	715							
amc$label_exit_name	1168	639							
amc$label_options	1170	641							
amc$label_type	1171	643							
amc$line_number	1172	645							
amc$loading_factor	1198	717							
amc$lock_expiration_time	1199	719							
amc$log_residence	1201	723							
amc$logging_options	1200	721							
amc$max_attribute	1242	1246							
amc$max_block_length	1173	647							
amc$max_block_number	755	758							
amc$max_error_count	1126	1129							
amc$max_file_id_ordinal	1063	1070							
amc$max_home_blocks	933	936							
amc$max_index_level	928	931							
amc$max_key_length	1267	1271							
amc$max_key_position	1276	1273							
amc$max_line_number	946	949							
amc$max_lines_per_inch	1334	1331							
amc$max_page_width	898	901							
amc$max_path_name_size	912	915							
amc$max_record_length	1174	649							
amc$max_records_per_block	1318	1322							
amc$max_statement_id_length	993	996							
amc$max_user_info	1329	1325							
amc$maximum_block	763	760	1301						
amc$maximum_keyed_record	1279	1276							
amc$maximum_record	969	972	1044	1305					
amc$message_control	1175	725							
amc$min_block_length	1176	651							
amc$min_record_length	1177	653							
amc$null_attribute	1178	655							
amc$open_position	1179	657							
amc$padding_character	1180	659							
amc$page_format	1181	661							
amc$page_length	1182	663							
amc$page_width	1183	665							
amc$preset_value	1185	667							
amc$record_limit	1186	727							
amc$record_type	1187	669							
amc$records_per_block	1188	729							
amc$return_option	1189	671							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
amc$ring_attributes	1190	673							
amc$sequence_pointer	17	24							
amc$statement_identifier	1191	675							
amc$user_info	1192	677							
amc$vertical_print_density	1193	679							
amt$access_selection	600	598							
amt$average_record_length	1044	687							
amt$block_header_type	736	739	745						
amt$block_number	758	741	748						
amt$block_status	737	750							
amt$block_type	1047	614							
amt$collation_value	1052	1049							
amt$compression_procedure_name	903	692							
amt$data_padding	1055	694							
amt$dynamic_home_block_space	922	696							
amt$entry_point_reference	906	903	924						
amt$error_limit	1122	700							
amt$estimated_record_count	1131	702							
amt$file_attribute_keys	1246	605							
amt$file_byte_address	12	410	411	451	455	467	469	485	1741
		1789	1790	1816	1840	2007	3760	5361	5362
		5388	5403	5467	5474	5487	5512	5512	7579
		8536	9075	9388	9508	9512			
amt$file_contents	781	626							
amt$file_id_ordinal	1070	1067							
amt$file_id_sequence	1071	1068							
amt$file_identifier	1066	1058	1285						
amt$file_item	604	600	603						
amt$file_limit	438	473	628	1663	2011	2888			
amt$file_organization	1252	630							
amt$file_position	1255	591							
amt$file_processor	842	632							
amt$file_structure	888	634							
amt$forced_write	1258	636							
amt$hashing_procedure_name	924	704							
amt$index_levels	931	706							
amt$index_padding	1262	708							
amt$initial_home_block_count	936	710							
amt$internal_code	1264	638							
amt$key_length	1271	712							
amt$key_position	1273	714							
amt$key_type	1282	716							
amt$label_options	593	642							
amt$label_type	1289	644							
amt$line_number	940	646							
amt$line_number_length	949	941							
amt$line_number_location	951	942							
amt$loading_factor	954	718							
amt$lock_expiration_time	956	720							
amt$log_residence	958	724							
amt$logging_options	961	722							
amt$logging_possibilities	964	961							
amt$max_block_length	760	648	740	746	747				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
amt$max_record_length	972	650							
amt$message_control	1299	726							
amt$min_block_length	1301	652							
amt$min_record_length	1305	654							
amt$open_position	976	658							
amt$padding_character	1308	660							
amt$page_format	891	662							
amt$page_length	895	664							
amt$page_width	901	666	951	998					
amt$path_name	915	908	958						
amt$pointer_kind	16	19							
amt$preset_value	442	493	668	1661	2898				
amt$record_limit	1310	728							
amt$record_type	1314	670							
amt$records_per_block	1322	730							
amt$return_option	594	672							
amt$ring_attributes	980	674							
amt$statement_id_length	996	988							
amt$statement_id_location	998	989							
amt$statement_identifier	987	676							
amt$tape_error_action	1008	1003							
amt$tape_error_options	1001	622							
amt$unused_bit_count	768	742	749						
amt$user_info	1325	678							
amt$vertical_print_density	1331	680							
asid	3862	7832	7833	7834	7835				
asid	4473	7351	7356/M	7835/M	8206/M	8598/M	8706/M	10036/M	
asid	7133	7138/M	7141/S	7142/M	7142				
asid	7149	7155/M	7157/M	7157	7157				
asid	7324	7355/M	7355/M	7355	7355				
asid	7344	7355/P	7356						
assign_active	4500	7598	7652/M	8216/M	9543				
assign_active	7579	7598/M	7607/M	7624	7640/P				
assign_active_sfid	4103	7654/M	9534	9535/P	9537/P				
aste_p	7343	7358/M	7360	7360	7361	7363	7363	7366	7366
		7367							
asti	2006	7173	7350	9012					
asti	4442	7354/M	7358/S	8207/M					
asti	7135	7139/M	7141/M	7141	7144				
asti	7152	7154/M	7155/S	7155/S	7155/S	7156/S			
asti	7324	7355/M	7355/S	7355/S	7355/S	7355/S			
asti	7345	7350/M	7351	7354	7355/P				
attach_count	2001	7365	7369	7384	8225/M	8687/M			
attrib	7663	7692	7693	7694	7695	7706	7716	7728	7729
		7729	7730	7733	7733	7734	7738	7739	7741
		7743	7748	7749	7756	7761	7766	7769	7772
		7773	7782	7783	7784	7785	7786	7787	7792
		7798	7800	7803	7805	7811	7812	7816	7819
		7820	7824	7825	7827	7832	7833	7834	7835
		7843	7845	7847	7851	7856	7857	7859/P	7869
		7873	7915	7944					
bits	7127	7141	7155	7155	7155	7156	7355	7355	7355

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7355							
byte_offset	3774	9253/M	9257						
byte_offset	9241	9257/M							
bytes_to_allocate	9384	9404/M	9405						
cache_bypass	3923	7805	8508/M	8510/M	9872				
cache_bypass	7455	7476/M	7487/M	7492/M	7550				
caller_id	8856	8884	8890						
caller_ring	8029	8037/S							
cell_p	7456	7522/M	7524	7551/M	7552				
cell_pointer	3905	7952/M							
child_sdt_entry_p	8127	8249/M	8251/M						
child_sdtx_entry_p	8128	8250/M	8252/M	8259/M	8260/M				
child_xcb_p	8129	8150/P	8151	8191/M	8192/M	8193/M	8194/M	8195/M	8249/P
		8250/P	8259/P						
class	459	7619/M	7621/M						
clear_space	3849	8505/M							
cnv	5572	5577/M	5577						
cnv	5589	5622/M	5623						
cnv	5706	5710/M	5710						
cnv	5715	5746/M	5746						
cnv	5759	5771/M	5772						
cnv	5789	5797/M	5797						
cnv	7447	7514/M	7514						
cnv	7447	7516/M	7516						
cnv	7572	7612/M	7612						
cnv	7572	7645/M	7645						
cnv	7662	7712/M	7712	7717/M	7717				
cnv	7662	7914/M	7914	7955/M	7955				
cnv	7992	8022/M	8022						
cnv	8074	8094/M	8094						
cnv	8121	8243/M	8243						
cnv	8121	8245/M	8245						
cnv	8390	8417/M	8417						
cnv	8424	8520/M	8520						
cnv	8526	8552/M	8552						
cnv	8526	8555/M	8555						
cnv	8573	8599/M	8599						
cnv	8649	8711/M	8711						
cnv	8849	8896/M	8896	8907/M	8907	8929/M	8929	8952/M	8952
cnv	8849	8926/M	8926						
cnv	9295	9338/M	9338						
cnv	9295	9340/M	9340						
cnv	9507	9535/M	9535						
cnv	9507	9538/M	9538						
cnv	9781	9817/M	9817						
cnv	9781	9819/M	9819						
cnv	9985	10009/M	10009	10019/M	10019	10028/M	10028		
code	5692	5577/M	5620/M	5710/M	5746/M	5770/M	5797/M	7514/M	7516/M
		7612/M	7645/M	7712/M	7717/M	7914/M	7955/M	8022/M	8094/M
		8243/M	8245/M	8417/M	8520/M	8552/M	8555/M	8599/M	8711/M
		8896/M	8907/M	8926/M	8929/M	8952/M	9338/M	9340/M	9535/M
		9538/M	9817/M	9819/M	10009/M	10019/M	10028/M		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
code	5706	5710							
code	5715	5746							
code	5754	5770							
code	7447	7516							
code	7572	7645							
code	7662	7914	7955						
code	7992	8022							
code	8074	8094							
code	8121	8245							
code	8424	8520							
code	8526	8555							
code	8573	8599							
code	8649	8711							
code	8849	8896	8907	8929	8952				
code	9295	9340							
code	9507	9538							
code	9781	9819							
code	9985	10009	10019	10028					
condition	1077	7899	9091	9091	9092	9412	9552	9553	9554
		9554	9555	9556	9562				
condition	3571	6120/P	7264/P	8362/P	8795	9258/P			
content	8982	8744	8745						
count	2066	5577/M	5577	5607/M	5607	5710	5710/M	5710	5732
		5733/M	5733	5797/M	5797	7514/M	7514	7516	7516/M
		7516	7612/M	7612	7645	7645/M	7645	7712/M	7712
		7717/M	7717	7914	7914/M	7914	7955	7955/M	7955
		8022	8022/M	8022	8094	8094/M	8094	8243/M	8243
		8245	8245/M	8245	8417/M	8417	8520	8520/M	8520
		8552/M	8552	8555	8555/M	8555	8599	8599/M	8599
		8711	8711/M	8711	8896	8896/M	8896	8907	8907/M
		8907	8926/M	8926	8929	8929/M	8929	8952	8952/M
		8952	9338/M	9338	9340	9340/M	9340	9535/M	9535
		9538	9538/M	9538	9817/M	9817	9819	9819/M	9819
		10009	10009/M	10009	10019	10019/M	10019	10028	10028/M
		10028							
count	8785	8792	8809						
create_sdt_and_sdtx	8199	8199	8262						
ctime	9076	9093/M	9097						
current_length	3843	8446/M							
current_time	9385	9409/M	9415						
cycle_rb	5572	5577/M	5577/M	5577/M	5577/M	5577/M	5577/P	5577/P	
cycle_rb	5598	5619/M	5620/M	5621/M	5623/M	5624/M	5625/P	5625/P	
cycle_rb	5789	5797/M	5797/M	5797/M	5797/M	5797/M	5797/P	5797/P	
cycle_rb	7447	7514/M	7514/M	7514/M	7514/M	7514/M	7514/P	7514/P	
cycle_rb	7572	7612/M	7612/M	7612/M	7612/M	7612/M	7612/P	7612/P	
cycle_rb	7662	7712/M	7712/M	7712/M	7712/M	7712/M	7712/P	7712/P	7717/M
		7717/M	7717/M	7717/M	7717/M	7717/P	7717/P		
cycle_rb	8121	8243/M	8243/M	8243/M	8243/M	8243/M	8243/P	8243/P	
cycle_rb	8390	8417/M	8417/M	8417/M	8417/M	8417/M	8417/P	8417/P	
cycle_rb	8526	8552/M	8552/M	8552/M	8552/M	8552/M	8552/P	8552/P	
cycle_rb	8849	8926/M	8926/M	8926/M	8926/M	8926/M	8926/P	8926/P	
cycle_rb	9295	9338/M	9338/M	9338/M	9338/M	9338/M	9338/P	9338/P	
cycle_rb	9507	9535/M	9535/M	9535/M	9535/M	9535/M	9535/P	9535/P	

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cycle_rb	9781	9817/M	9817/M	9817/M	9817/M	9817/M	9817/P	9817/P	
cycle_task	5706	5710/M	5710/M	5710/M	5710/M	5710/M	5710/P	5710/P	
cycle_task	5715	5746/M	5746/M	5746/M	5746/M	5746/M	5746/P	5746/P	
cycle_task	5767	5769/M	5770/M	5772/M	5773/M	5774/M	5775/P	5775/P	
cycle_task	7447	7516/M	7516/M	7516/M	7516/M	7516/M	7516/P	7516/P	
cycle_task	7572	7645/M	7645/M	7645/M	7645/M	7645/M	7645/P	7645/P	
cycle_task	7662	7914/M	7914/M	7914/M	7914/M	7914/M	7914/P	7914/P	7955/M
		7955/M	7955/M	7955/M	7955/M	7955/P	7955/P		
cycle_task	7992	8022/M	8022/M	8022/M	8022/M	8022/M	8022/P	8022/P	
cycle_task	8074	8094/M	8094/M	8094/M	8094/M	8094/M	8094/P	8094/P	
cycle_task	8121	8245/M	8245/M	8245/M	8245/M	8245/M	8245/P	8245/P	
cycle_task	8424	8520/M	8520/M	8520/M	8520/M	8520/M	8520/P	8520/P	
cycle_task	8526	8555/M	8555/M	8555/M	8555/M	8555/M	8555/P	8555/P	
cycle_task	8573	8599/M	8599/M	8599/M	8599/M	8599/M	8599/P	8599/P	
cycle_task	8649	8711/M	8711/M	8711/M	8711/M	8711/M	8711/P	8711/P	
cycle_task	8849	8896/M	8896/M	8896/M	8896/M	8896/M	8896/P	8896/P	8907/M
		8907/M	8907/M	8907/M	8907/M	8907/P	8907/P	8929/M	8929/M
		8929/M	8929/M	8929/M	8929/P	8929/P	8952/M	8952/M	8952/M
		8952/M	8952/M	8952/P	8952/P				
cycle_task	9295	9340/M	9340/M	9340/M	9340/M	9340/M	9340/P	9340/P	
cycle_task	9507	9538/M	9538/M	9538/M	9538/M	9538/M	9538/P	9538/P	
cycle_task	9781	9819/M	9819/M	9819/M	9819/M	9819/M	9819/P	9819/P	
cycle_task	9985	10009/M	10009/M	10009/M	10009/M	10009/M	10009/P	10009/P	10019/M
		10019/M	10019/M	10019/M	10019/P	10019/P	10028/M	10028/M	
		10028/M	10028/M	10028/M	10028/P	10028/P			
delay_status	9386	9413/P							
dest	5819	5829							
dest	9111	9144							
destination_pva	3773	9252/M							
destination_pva	4621	9143/P							
destination_pva	9240	9252							
destroy_segment	7161	7184	8933	8959	9954	10031			
dfc$command_record_bytes	6345	6353							
dfc$division_overwrite_words	6332	6360							
dfc$esm_command_record_size	6353	6361							
dfc$esm_header_record_size	6354	6361							
dfc$esm_maintenance_buf_size	6333	6364							
dfc$esm_memory_base_shift	6339	6361	6362	6362					
dfc$header_record_bytes	6344	6354							
dfc$max_esm_memory_size	6334	6363							
dfc$max_number_of_mainframes	6341	6326							
dfc$min_cdcnet_errors	327	333	335	338	341	344	347	350	353
		356	359	362	382				
dfc$min_data_record_bytes	6349	6360							
dfc$min_driver_test_errors	322	370	373	376	379				
dfc$min_ecc	32	38	41	44	47	50	54	58	61
		64	67	70	73	76	79	82	85
		88	91	95	98	101	105	108	111
		114	117	120	123	126	129	132	135
		138	141	144	148	152	155	158	162
		165	168	171	174	177	180	183	186
		190	193	197	200	203	206	209	212

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		215	219	223	227	230	233	236	240
		244	247	250	253	256	259	263	266
		270	273	276	279	282	285	289	292
		295	299	303	308	311	315	322	324
		327	329						
dfc$min_esm_division_size	6359	6363							
dfc$min_mm_recovery_errors	329	385	389	392					
dfd_p	5409	5423/M							
dfd_p	8526	8553/M							
dfd_p	8537	8553/P	8556						
dfe$family_not_served	197	9091	9554						
dfe$server_has_terminated	180	9092	9555						
dfe$server_not_active	129	9091	9554						
dft$mainframe_set	6326	6276	6277	6397	6398				
disk_file_descriptor_p	2021	5422	8553						
dmc$a2	1351	1355							
dmc$allocated_length	512	450	1839						
dmc$asid	512	452							
dmc$byte_address	513	454							
dmc$bytes_per_allocation	513	456							
dmc$bytes_per_level_2	1772	1780							
dmc$chapter_length	520	504	2905						
dmc$class	513	458	1541	2881	7617				
dmc$class_ordinal	513	460	1543	2883					
dmc$clear_space	514	462	2885						
dmc$default_number_fau_entries	1984	1977							
dmc$device_file_list_index	514	464	1841						
dmc$device_manager_error_code	2223	2224	2227	2230	2233	2236	2239	2242	2245
		2248	2251	2254	2257	2260	2263	2266	2269
		2272	2275	2278	2281	2284	2287	2290	2293
		2296	2299	2302	2305	2308	2311	2314	2317
		2320	2323	2326	2329	2332	2335	2338	2341
		2344	2347	2350	2353	2356	2359	2362	2365
		2368	2371	2374	2377	2380	2383	2386	2389
		2392	2395	2398	2401	2404	2407	2410	2413
		2416	2419	2422	2425	2428	2431	2434	2437
		2440	2443	2446	2449	2452	2455	2458	2461
		2464	2467	2470	2473	2476	2479	2482	2485
		2488	2491	2494	2497	2500	2503	2506	2509
		2512	2515	2518	2521	2524	2527	2530	2533
		2536	2539	2542	2545	2548	2551	2554	2557
		2560	2563	2566	2569	2572	2575	2578	2581
		2584	2587	2590	2593	2596	2599	2602	2605
		2608	2611	2614	2617	2620	2623	2626	2629
		2632	2635	2638	2641	2644	2647	2650	2653
		2656	2659	2662	2665	2668	2671	2674	2677
		2680	2683	2686	2689	2692	2695	2698	2701
		2704	2707	2710	2713	2716	2719	2722	2725
		2728	2731	2734	2737	2740	2743	2746	2749
		2752	2755	2758	2761	2764	2767	2770	2773
		2776	2779	2782	2785	2791	2794	2797	2800
		2803	2806	2809	2812	2815	2818	2821	2824
		2827	2830	2834	2837	2840	2845	2848	2851

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		2854	2857	2860	2863	2866	2869	2872
dmc$eof_byte_address	514	466						
dmc$eoi_byte_address	515	468						
dmc$file_hash	515	470						
dmc$file_kind	516	476	1547					
dmc$file_limit	515	472	2887					
dmc$file_status	515	474						
dmc$global_file_name	516	478						
dmc$internal_vsn	516	480	1843					
dmc$level_1_table_size	1768	1772	1775					
dmc$locked_file	516	482	2889					
dmc$logical_length	517	484						
dmc$master_volume_required	517	486	1549	2891				
dmc$max_bytes_per_allocation	1347	406	1340	1342	1343			
dmc$max_bytes_per_dau	1881	1873						
dmc$max_bytes_per_mau	1917	1903						
dmc$max_class_ordinal	1372	1369						
dmc$max_dau_address	1883	1877	1925					
dmc$max_daus_allocation	1885	1874						
dmc$max_daus_position	1887	1875	1927					
dmc$max_daus_transfer	1889	1876						
dmc$max_device_file_list_index	1390	1387						
dmc$max_fau_entries	1985	1978						
dmc$max_file_hash	1400	432	434	1397				
dmc$max_mau_address	1925	1913						
dmc$max_maus_per_allocation	1919	1905	1906					
dmc$max_maus_per_dau	1921	1907	1908	1925	1927			
dmc$max_maus_per_transfer	1923	1910	1911					
dmc$max_maus_position	1927	1912						
dmc$max_transfer_size	1689	1684						
dmc$min_bytes_per_allocation	1346	1341						
dmc$min_bytes_per_dau	1880	1873						
dmc$min_dau_address	1882	1877						
dmc$min_mau_address	1924	1913						
dmc$min_maus_per_allocation	1918	1904						
dmc$min_maus_per_dau	1920	1907						
dmc$min_maus_per_transfer	1922	1909						
dmc$overflow	517	488	2893					
dmc$owner	518	490	2895					
dmc$preset_value	518	492	2897					
dmc$queue_status	520	508						
dmc$recorded_vsn	518	494	1545	1845				
dmc$requested_allocation_size	518	496	1551	2899				
dmc$requested_transfer_size	519	498	2901					
dmc$requested_volume	519	500	2903					
dmc$setname	519	502	1553					
dmc$write_mode	520	506						
dme$unable_to_alloc_all_space	2710	9412	9552	9556				
dme$unable_to_create_fdt_entry	2308	9562						
dme$unable_to_get_fd_lock	2332	9553						
dmp$allocate_file_space_r1	5359	7638						
dmp$create_disk_file	5385	7623						
dmp$destroy_file	5395	7181	9020					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dmp$fetch_eoi	5401	9399						
dmp$get_disk_file_descriptor_p	5408	8553						
dmp$get_initialized_addresses	5447	8566						
dmp$get_total_allocated_length	5465	8554	9558					
dmp$reallocate_file_space	5480	8812						
dmp$sparse_allocate	5485	7640						
dmt$access_kind	1521	1527						
dmt$active_volume_table_index	1858	1812						
dmt$addr_length_pair	5457	5449	8530					
dmt$allocation_size	1340	414	457	497	1552	1667	2900	
dmt$allocation_styles	1351	1824						
dmt$bytes_per_mau	1903	1817						
dmt$chapter_number	5374	5363						
dmt$class_member	1364	415	459	1363	1542	2882		
dmt$class_ordinal	1369	416	461	1544	2884			
dmt$dau_address	1877	1971						
dmt$daus_per_allocation	1874	1819						
dmt$daus_per_position	1875	1818						
dmt$delete_count	427	403						
dmt$delete_logging_count	1828	1814						
dmt$device_file_list_index	1387	465	1813	1842				
dmt$disk_file_descriptor	401	5409	8537	9387				
dmt$fau_states	1979	1972						
dmt$file_allocation_unit	1970	1800	1801	1968				
dmt$file_attribute	448	5387	7581					
dmt$file_attribute_keywords	512	449	1540	1838	2880			
dmt$file_hash	1397	471						
dmt$file_hash_thread	430	432						
dmt$file_medium_descriptor	1809	420	1823					
dmt$fmd_attribute	1837	1833						
dmt$fmd_index	1897	409	419	1832	1973			
dmt$global_file_name	1403	479	1666					
dmt$internal_vsn	1514	481	1820	1844				
dmt$level_1_index	1775	408	1789					
dmt$level_1_table	1789	407						
dmt$level_2_index	1780	1800						
dmt$locked_file	1524	483	2890					
dmt$maus_per_dau	1907	1821						
dmt$maus_per_transfer	1909	1822						
dmt$ms_overflow_allowed	1738	2894						
dmt$queue_status	1677	1664						
dmt$requested_volume	1557	418	501	2904				
dmt$system_file_id	1942	1811	3742	3746	3747	3755	3759	5360 5396
		5402	5447	5473	5480	6308		
dmt$transfer_size	1684	417	499	1668	2902			
dmt$usage_count	1696	1665						
dmt$write_lock	1523	1529						
dmv$idle_system	6135	7614						
enable	9034	9051						
eoi	9388	9399/P	9404					
eoi_byte_address	2007	5524	7632	8446	8561	8628		
eoi_state	2008	5518	8446	8628				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
err_exit_proc	3853	8499/M							
execute_privilege	3924	7811	7816	8512/M	9868	9878	9883		
expand_segment_table	7187	7285	7311	7902	9623				
fde_entry_p	7163	7172	7173	7179/P					
fde_entry_p	7328	7350	7364	7365	7365	7368	7369	7369	7384
		7384							
fde_entry_p	7450	7484	7485	7491	7510/P	7511/M	7511	7526/M	7544/P
		7545/M	7545						
fde_entry_p	7670	7707/P	7708	7916/P					
fde_entry_p	7999	8005/P	8018/P	8022/P					
fde_entry_p	8079	8086/P	8093/M	8093	8094/P				
fde_entry_p	8130	8221/P	8222	8224/M	8225/M	8226/M	8227/M	8228/M	8230/M
		8243/P	8244/M	8244	8245/P				
fde_entry_p	8431	8437/P	8446/P	8448	8449	8501	8520/P		
fde_entry_p	8538	8552/P	8553/P	8554/P	8555/P	8561			
fde_entry_p	8581	8587/P	8592/M	8599/P					
fde_entry_p	8857	8885/P	8896/P	8907/P	8919/M	8919	8950	8951/M	8952/P
		8957	8958	8959/P					
fde_entry_p	8985	9006/P	9011	9012	9019/M				
fde_entry_p	9305	9338/P	9339/M	9339	9340/P	9356/P			
fde_entry_p	9790	9817/P	9818/M	9819/P					
fde_entry_p	9925	9949/P	9950	9953/M	9953	9954/P			
fde_entry_p	9990	9999/P	10009/P	10019/P	10026/M	10026	10027	10028/P	10031/P
fde_locked	8672	8672	8677	8683	8709				
fde_p	3792	5521/M	8446/M	8628/M	9770/P				
fde_p	5408	5417	5420						
fde_p	5512	5518	5521	5524					
fde_p	5534	5549/M	5550	5551/M					
fde_p	5572	5577/P							
fde_p	5706	5710/P							
fde_p	5789	5795/M	5795	5795/M					
fde_p	5790	5795/P	5796	5797/P					
fde_p	7447	7514/P	7514	7514/P					
fde_p	7447	7514/M	7514	7514/M					
fde_p	7447	7516/P							
fde_p	7572	7612/P	7612	7612/P					
fde_p	7572	7612/M	7612	7612/M					
fde_p	7572	7645/P							
fde_p	7580	7612/P	7615	7623/P	7626/P	7632	7634	7645/P	
fde_p	7662	7712/P	7712	7712/P	7717/P	7717	7717/P		
fde_p	7662	7712/M	7712	7712/M	7717/M	7717	7717/M		
fde_p	7662	7914/P	7955/P						
fde_p	7992	8022/P							
fde_p	8074	8094/P							
fde_p	8121	8243/P	8243	8243/P					
fde_p	8121	8243/M	8243	8243/M					
fde_p	8121	8245/P							
fde_p	8390	8417/P	8417	8417/P					
fde_p	8390	8417/M	8417	8417/M					
fde_p	8424	8446	8446	8446					
fde_p	8424	8520/P							
fde_p	8526	8552/P	8552	8552/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
fde_p	8526	8552/M	8552	8552/M					
fde_p	8526	8553	8553						
fde_p	8526	8555/P							
fde_p	8573	8599/P							
fde_p	8604	8627/M	8627	8627/M					
fde_p	8604	8628	8628	8628					
fde_p	8611	8627/P	8628/P						
fde_p	8649	8711/P							
fde_p	8656	8664/P	8701	8711/P					
fde_p	8849	8896/P	8907/P	8929/P	8952/P				
fde_p	8849	8926/P	8926	8926/P					
fde_p	8849	8926/M	8926	8926/M					
fde_p	8982	9006/M	9006	9006/M					
fde_p	9295	9338/P	9338	9338/P					
fde_p	9295	9338/M	9338	9338/M					
fde_p	9295	9340/P							
fde_p	9507	9535/P	9535	9535/P					
fde_p	9507	9535/M	9535	9535/M	9545/M	9545	9545/M	9557/M	9557
		9557/M							
fde_p	9507	9538/P							
fde_p	9513	9535/P	9536	9538/P	9545/P	9546	9557/P	9558/P	
fde_p	9738	9770/M	9770	9770/M					
fde_p	9781	9817/P	9817	9817/P					
fde_p	9781	9817/M	9817	9817/M					
fde_p	9781	9819/P							
fde_p	9920	9949/M	9949	9949/M					
fde_p	9985	10009/P	10019/P	10028/P					
file_attributes_p	7581	7616	7617/M	7619/M	7621/M	7623/P			
file_entry_index	1947	5549	5795	7514	7612	7712	7717	8243	8417
		8552	8627	8926	9006	9338	9535	9545	9557
		9770	9817	9949					
file_hash	1949	5550	5795	7514	7612	7712	7717	8223/M	8243
		8417	8552	8627	8926	9006	9338	9535	9545
		9557	9770	9817	9949				
file_hash	2004	5550	5795	7514	7612	7712	7717	8227/M	8243
		8417	8552	8627	8926	9006	9338	9535	9545
		9557	9770	9817	9949				
file_kind	2003	7484	7714/M	8226/M	8931	8932	8957	8958	
file_limit	2011	7798/M	7843/M	7938	7939	8448	8449	8701/M	8701
		9818/M							
file_limits	7582	7597/M	7606/M	7638/P	7640/P				
file_limits_enforced	4498	7597	7695/M	8595/M	8959/P	9020/P	9955/P	10031/P	
file_limits_enforced	7164	7181/P							
file_limits_to_enforce	3804	7695							
find_available_segment_number	7289	7320	7896	9278	9321				
flags	1997	8592/M							
flush_pages	9514	9546/M	9549						
gfc$fde_size	5566	5549	5795	7514	7612	7712	7717	8243	8417
		8552	8627	8926	9006	9338	9535	9545	9557
		9770	9817	9949					
gfc$fde_table_base	5564	5549	5565	5795	7514	7612	7712	7717	8243
		8417	8552	8627	8926	9006	9338	9535	9545

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9557	9770	9817	9949				
gfc$fk_catalog	1576	1588							
gfc$fk_global_unnamed	1580	8932	8958						
gfc$fk_job_local_file	1578	1587							
gfc$fk_unnamed_file	1579	7484	7714	8226	8931	8957			
gfc$fm_mass_storage_file	2058	2020	5417	8553					
gfc$fm_served_file	2059	2023	7634						
gfc$fm_transient_segment	2058	7172	7615	9011	9546				
gfc$qs_global_shared	1753	7368	7491						
gfc$qs_job_shared	1753	7365	7369	7384					
gfc$qs_job_working_set	1753	7364	7485	7713					
gfc$tr_job	1960	5543	5795	7361	7514	7612	7707/P	7712	7717
		8221/P	8243	8417	8552	8627	8680/P	8926	9006
		9338	9535	9545	9557	9770	9817	9949	
gfc$tr_system	1960	5542	5795	7362	7514	7612	7712	7717	8243
		8417	8552	8627	8926	9006	9338	9535	9545
		9557	9770	9817	9949				
gfp$assign_fde	5495	7707	8221	8680					
gfp$clear_signature_lock	5715	5710	5750	7516	7645	7914	7955	8022	8094
		8245	8520	8555	8599	8711	8896	8907	8929
		8952	9340	9538	9819	10009	10019	10028	
gfp$free_fde	5505	7179	7916						
gfp$get_eoi_from_fde	5511	5526	8446	8628					
gfp$get_eoi_from_fde	5512	5524/M	8446/M	8628/M					
gfp$get_fde_p	5533	5554	5795	7514	7612	7712	7717	8243	8417
		8552	8627	8926	9006	9338	9535	9545	9557
		9770	9817	9949					
gfp$get_locked_fde_p	5789	5800	7514	7612	7712	7717	8243	8417	8552
		8926	9338	9535	9817				
gfp$lock_fde	5572	5578							
gfp$set_signature_lock	5582	5577	5629	5797	7514	7612	7712	7717	8243
		8417	8552	8926	9338	9535	9817		
gfp$unlock_fde_p	5706	5712	7516	7645	7914	7955	8022	8094	8245
		8520	8555	8599	8711	8896	8907	8929	8952
		9340	9538	9819	10009	10019	10028		
gft$allocation_unit_size	2044	2009							
gft$attach_count	2049	2000	2001						
gft$fde_flags	2029	1997							
gft$file_desc_entry_p	2912	3792	5386	5408	5499	5506	5512	5534	5572
		5790	7163	7670	7683	8611	8657	9925	
gft$file_descriptor_entry	1994	430	1999	2912	2916	5414			
gft$file_descriptor_index	1957	1947							
gft$file_kind	1572	477	1548	1584	2003				
gft$file_media	2058	2019							
gft$locked_file_desc_entry_p	2916	5466	5706	7328	7450	7463	7580	7672	7999
		8079	8130	8394	8431	8538	8581	8656	8857
		8867	8985	9305	9513	9790	9990		
gft$open_count	2088	2002	2093	8868	9991				
gft$queue_status	1753	509	2012						
gft$segment_lock_info	2092	2005							
gft$signature_lock	2064	1995	5583	5716					
gft$system_file_identifier	1946	1942	3591	3768	3806	4104	4491	4583	5352
		5389	5486	5498	5533	5789	6138	6799	7162

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7169	7326	7574	7588	7682	8144	8310	8327
		8658	8787	8869	9239	9296			
gft$table_residence	1960	1948	5496						
gft$transfer_unit_size	2055	2010							
gft$trick_pointer	5430	5415							
gfv$null_sfid	6138	7654	7706	7915	9534	9547/P			
gl_key	3847	7800	8454/M						
global_task_id	2016	7922/M							
global_task_id	4065	5577	5604	5710	5726	5797	6050	7514	7516
		7612	7645	7712	7717	7914	7922	7955	8022
		8094	8243	8245	8417	8520	8552	8555	8599
		8711	8896	8907	8926	8929	8952	9099	9180
		9338	9340	9535	9538	9817	9819	10009	10019
		10028							
global_task_id	6033	6050/M							
global_task_id	9072	9099/M							
global_task_id	9149	9180/M							
global_template_file	2033	8592/M							
gtid	2067	5577	5577/M	5577	5577	5606	5614/M	5621	5621
		5710	5710/M	5728	5735/M	5797	5797/M	5797	5797
		7514	7514/M	7514	7514	7516	7516/M	7612	7612/M
		7612	7612	7645	7645/M	7712	7712/M	7712	7712
		7717	7717/M	7717	7717	7914	7914/M	7955	7955/M
		8022	8022/M	8094	8094/M	8243	8243/M	8243	8243
		8245	8245/M	8417	8417/M	8417	8417	8520	8520/M
		8552	8552/M	8552	8552	8555	8555/M	8599	8599/M
		8711	8711/M	8896	8896/M	8907	8907/M	8926	8926/M
		8926	8926	8929	8929/M	8952	8952/M	9338	9338/M
		9338	9338	9340	9340/M	9535	9535/M	9535	9535
		9538	9538/M	9817	9817/M	9817	9817	9819	9819/M
		10009	10009/M	10019	10019/M	10028	10028/M		
gtid	9077	9099/P	9100/P						
gtid	9152	9180/P	9181/P						
hardware_attri_set	3856	7743	7748	7756	7761	7766	7769	7772	8456/M
		8459/M	8459	8462/M	8462	8465/M	8465	8471/M	8471
		8474/M	8474	8480/M	8480	8483/M	8483	8486/M	8486
		8492/M	8492	9824	9829	9834	9842	9847	9852
		9855	9858						
heap_pointer	3909	7950/P							
highest_offset_allocated	410	8556							
i	5572	5577/M							
i	5592	5622/M							
i	5706	5710/M							
i	5715	5746/M							
i	5762	5771/M							
i	5789	5797/M							
i	7134	7140							
i	7447	7514/M							
i	7447	7516/M							
i	7457	7537	7538/S						
i	7572	7612/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
i	7572	7645/M						
i	7662	7712/M	7717/M					
i	7662	7914/M	7955/M					
i	7669	7698	7699/S	7729	7730/S	7733/S	7733/S	7734/S	7738/S
		7739/S	7741/S	7743/S	7748/S	7756/S	7761/S	7766/S	7769/S
		7772/S	7783/S	7785/S	7787/S	7792/S	7798/S	7800/S	7803/S
		7805/S	7811/S	7816/S	7819/S	7824/S	7825/S	7832/S	7833/S
		7834/S	7835/S	7843/S	7845/S	7847/S	7851/S	7856/S	7857/S
		7859/S	7869/S	7873/S					
i	7992	8022/M						
i	8074	8094/M						
i	8121	8243/M						
i	8121	8245/M						
i	8320	8379	8380/S	8380/S					
i	8390	8417/M						
i	8424	8520/M						
i	8432	8438	8439/S	8441/S	8442/S	8444/S	8446/S	8448/S	8449/S
		8451/S	8451/S	8454/S	8456/S	8459/S	8459/S	8462/S	8462/S
		8465/S	8465/S	8471/S	8471/S	8474/S	8474/S	8480/S	8480/S
		8483/S	8483/S	8486/S	8486/S	8492/S	8492/S	8497/S	8499/S
		8501/S	8503/S	8505/S	8508/S	8510/S	8512/S	8513/S	8514/S
i	8526	8552/M						
i	8526	8555/M						
i	8573	8599/M						
i	8649	8711/M						
i	8733	8764	8765/P	8767/S	8773	8774/S		
i	8849	8896/M	8907/M	8929/M	8952/M			
i	8849	8926/M						
i	8858	8875	8876/S	8914	8915/S			
i	9120	9130	9132/P					
i	9295	9338/M						
i	9295	9340/M						
i	9507	9535/M						
i	9507	9538/M						
i	9607	9619	9638/S					
i	9781	9817/M						
i	9781	9819/M						
i	9791	9810	9811/S	9814/S	9815/S	9818/S	9824/S	9829/S	9834/S
		9842/S	9847/S	9852/S	9855/S	9858/S	9868/S	9872/S	9878/S
		9883/S	9886/S	9891/S	9892/S				
i	9985	10009/M	10019/M	10028/M					
i#build_adaptable_heap_pointer	5804	7949							
i#build_adaptable_seq_pointer	5810	7946							
i#call_monitor	5529	5522	5577	5625	5710	5746	5775	5797	7177
		7263	7390	7514	7516	7612	7645	7712	7717
		7914	7955	8022	8094	8243	8245	8361	8417
		8446	8520	8552	8555	8599	8628	8711	8793
		8896	8907	8926	8929	8952	9016	9255	9338
		9340	9535	9538	9550	9775	9817	9819	10009
		10019	10028						
i#move	5818	5834	9143						
i#program_error	5444	5418	8553						
i#real_memory_address	5836	8190	9131						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	6188	7361	7363						
ijl_ordinal	6792	7361	7363	7367					
ijle_p	6187	7486							
in_memory	9121	9127/M	9134/M	9139					
in_use	6793	7360							
index	2075	5577	5621	5797	7514	7612	7712	7717	8243
		8417	8552	8926	9338	9535	9817		
inheritance	3864	7845	8503/M						
inheritance	4492	7845/M	7980	7981/M	7982	7983/M	8160	8160	8202
		8220	8237	8238/M	8503	8593/M	8895	9049	9946
init_new_io	3720	8791/M	8816/M						
ioc$max_unit_number	1864	1858	1867						
iot$io_error	6860	6309	6835						
iot$transfer_count	4662	4650							
jmc$detached_job_wait_time_max	6216	6213							
jmc$highest_det_job_wait_time	6226	6216	6227						
jmc$highest_prio_age_interval	4349	4340	4350						
jmc$highest_service_accumulator	4381	4382							
jmc$highest_service_factor_valu	4423	4416							
jmc$highest_working_set_size	6923	6914	6924	6926	6928	6930			
jmc$ies_job_swapped	6429	6438							
jmc$ies_swapin_in_progress	6428	6437							
jmc$iss_idle_tasks_initiated	6444	6471							
jmc$iss_swapin_io_complete	6469	6472							
jmc$iss_swapin_requested	6465	6472							
jmc$iss_swapout_complete	6464	6471							
jmc$iss_swapped_io_cannot_init	6455	6482							
jmc$iss_swapped_no_io	6446	6481							
jmc$keyword_offset_maximum	4366	4341	6915						
jmc$kjl_maximum_entries	4314	4307	4308	6552					
jmc$kol_maximum_entries	4324	4309							
jmc$max_active_jobs	4305	4292	4300	4301					
jmc$max_ajl_ord	4306	4305	6373						
jmc$max_dispatching_control	4250	4254							
jmc$max_dispatching_priority	4151	4111	4114	4115					
jmc$maximum_job_classes	6541	6544							
jmc$maximum_job_count	4321	4314							
jmc$maximum_output_count	4331	4324							
jmc$maximum_service_classes	4399	4402							
jmc$min_dispatching_control	4249	4253							
jmc$null_service_class	4392	4393							
jmc$priority_aging_interval_max	4340	4337							
jmc$priority_p1	4165	4112							
jmc$priority_p10	4174	4113							
jmc$priority_p14	4178	4113							
jmc$priority_p8	4172	4112							
jmc$required_offset	4364	6929							
jmc$reserved_ajls	4310	4305							
jmc$service_accumulator_maximum	4373	4370							
jmc$service_factor_value_max	4416	4413							
jmc$system_default_offset	4365	4366	6931						
jmc$system_supplied_name_size	6714	6711							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmc$unlimited_offset	4362	4351	4383	6217	6228	6925		
jmc$unspecified_offset	4363	6927						
jmc$working_set_size_maximum	6914	6911						
jmt$ajl_ordinal	6373	6245						
jmt$delayed_swapin_work	6390	6275	6394					
jmt$detached_job_wait_time	6213	6198						
jmt$dispatching_control	4220	4203						
jmt$dispatching_control_index	4253	4220	6409					
jmt$dispatching_controls	4223	4221						
jmt$dispatching_priority	4111	4076	4078	4225	6257	6410	6411	6412
jmt$ijl_block_index	2205	2201						
jmt$ijl_block_number	2204	2200						
jmt$ijl_dispatching_control	6408	6258						
jmt$ijl_entry_status	6424	6244						
jmt$ijl_ordinal	2199	6188	6264	6292	6729	6730	6792	6827	6966
jmt$ijl_page_fault_count	6498	6493	6494	6495				
jmt$ijl_page_stats	6492	6488						
jmt$ijl_service_class_stats	6486	6279						
jmt$ijl_statistics	6516	6278						
jmt$ijl_swap_count	6507	6503	6504					
jmt$ijl_swap_counts	6502	6298	6489					
jmt$ijl_swap_status	6442	6247	6248	6249				
jmt$initiated_job_list_entry	6241	6187	6756					
jmt$input_file_location	6572	6567						
jmt$job_abort_disposition	6581	6565						
jmt$job_class	6544	6303						
jmt$job_control_block	6169	6155						
jmt$job_mode	6547	6260						
jmt$job_priority	4281	4212	4213	4214	4215	6300	6301	
jmt$job_recovery_disposition	6584	6566						
jmt$job_system_id	6903	6184						
jmt$kjl_index	6552	6246	6903					
jmt$maximum_active_jobs	4292	4197						
jmt$priority_aging_interval	4337	4205						
jmt$queue_file_ijl_information	6564	6285						
jmt$scheduling_data	6291	6269						
jmt$scheduling_priority	4211	4204						
jmt$service_accumulator	4370	4195	4196	6293	6294	6295		
jmt$service_class_index	4402	4188	4198	6304				
jmt$service_class_name	4405	4190	4191					
jmt$service_factor_value	4413	4199						
jmt$service_factors	4409	4199						
jmt$swap_data	6307	6271						
jmt$swapout_reasons	6589	6299						
jmt$swapped_job_entry	6604	6207	6316	6757				
jmt$system_supplied_name	6711	6182	6242					
jmt$task_time_slice	4263	4243	4244					
jmt$time_slice_values	4242	4089	4227					
jmt$user_supplied_name	6907	6183						
jmt$working_set_size	6911	6194	6195					
jmv$executing_within_system_job	6145	7618						
jmv$jcb	6155	7361	7363	7486				
jmv$system_ijl_ordinal	6966	7367						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmv$task_private_temp1_p	6971	8744	8745						
job_lock	1995	5577/P	5710/P	5797/P	7514/P	7516/P	7612/P	7645/P	7712/P
		7717/P	7914/P	7955/P	8022/P	8094/P	8243/P	8245/P	8417/P
		8520/P	8552/P	8555/P	8599/P	8711/P	8896/P	8907/P	8926/P
		8929/P	8952/P	9338/P	9340/P	9535/P	9538/P	9817/P	9819/P
		10009/P	10019/P	10028/P					
jsc$isqi_swapped_io_completed	6734	6736							
jsc$isqi_swapped_io_not_init	6733	6736							
jst$changed_asid_entry	6779	6770							
jst$ijl_swap_queue_id	6733	6728							
jst$ijl_swap_queue_link	6727	6253							
jst$io_control_information	6741	6272							
jst$swap_file_descriptor	6755	6273							
jst$swapped_page_descriptor	6764	6762							
jst$swapped_page_descriptors	6761	6758							
key_lock	4474	7800/M	8454						
keyword	449	7617/M							
keyword	3836	7730	8439	9811					
last_segment_number	2015	7921/M	8228/M						
leftover	8734	8755/M	8758/M	8772					
length	5459	8561/M							
length	5820	5827	5830/M	5830					
length	5940	5952	5952	5957/S					
length	8308	8338	8338	8338/S					
length	9111	9144	9144/M	9144					
length	9380	9404	9411/P						
length_to_allocate	7583	7632/M	7633	7635	7636/M	7638/P			
list_overflow	8532	8559/M	8567/P						
local_fde_p	5414	5420/M	5421	5422					
local_fde_p	8526	8553/M	8553	8553					
local_sdt	7458	7469/M	7478	7478	7479	7488/M	7493/M	7507/P	7510/P
		7522	7525/S	7526	7543/P	7544/P			
local_sdt_p	9926	9961/M	9963	9964/P	9964/P	9964/P	9968		
local_sdtp	8131	8167	8176/P	8177/P	8182/P	8182/P	8190/P	8194	
local_sdtx	7459	7470/M	7506/M	7507/P	7509	7510/P	7513	7514/P	7535/M
		7535	7542	7544/P					
local_sdtx_p	9927	9962/M	9967						
local_sdtxp	8132	8188	8189/P	8189/P	8195				
lock	5572	5577	5577	5577/M	5577	5577/M	5577	5577/M	5577/M
		5577	5577	5577					
lock	5583	5606	5606	5607/M	5607	5608/M	5612	5614/M	5615/M
		5621	5621	5624					
lock	5706	5710	5710	5710/M	5710	5710/M	5710	5710/M	5710/P
lock	5716	5728	5732	5733/M	5733	5735/M	5736	5737/M	5746/P
lock	5789	5797	5797	5797/M	5797	5797/M	5797	5797/M	5797/M
		5797	5797	5797					
lock	7447	7514	7514	7514/M	7514	7514/M	7514	7514/M	7514/M
		7514	7514	7514					
lock	7447	7516	7516	7516/M	7516	7516/M	7516	7516/M	7516/P
lock	7572	7612	7612	7612/M	7612	7612/M	7612	7612/M	7612/M
		7612	7612	7612					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	7572	7645	7645	7645/M	7645	7645/M	7645	7645/M	7645/P
lock	7662	7712	7712	7712/M	7712	7712/M	7712	7712/M	7712/M
		7712	7712	7712	7717	7717	7717/M	7717	7717/M
		7717	7717/M	7717/M	7717	7717	7717		
lock	7662	7914	7914	7914/M	7914	7914/M	7914	7914/M	7914/P
		7955	7955	7955/M	7955	7955/M	7955	7955/M	7955/P
lock	7992	8022	8022	8022/M	8022	8022/M	8022	8022/M	8022/P
lock	8074	8094	8094	8094/M	8094	8094/M	8094	8094/M	8094/P
lock	8121	8243	8243	8243/M	8243	8243/M	8243	8243/M	8243/M
		8243	8243	8243					
lock	8121	8245	8245	8245/M	8245	8245/M	8245	8245/M	8245/P
lock	8390	8417	8417	8417/M	8417	8417/M	8417	8417/M	8417/M
		8417	8417	8417					
lock	8424	8520	8520	8520/M	8520	8520/M	8520	8520/M	8520/P
lock	8526	8552	8552	8552/M	8552	8552/M	8552	8552/M	8552/M
		8552	8552	8552					
lock	8526	8555	8555	8555/M	8555	8555/M	8555	8555/M	8555/P
lock	8573	8599	8599	8599/M	8599	8599/M	8599	8599/M	8599/P
lock	8649	8711	8711	8711/M	8711	8711/M	8711	8711/M	8711/P
lock	8849	8896	8896	8896/M	8896	8896/M	8896	8896/M	8896/P
		8907	8907	8907/M	8907	8907/M	8907	8907/M	8907/P
		8929	8929	8929/M	8929	8929/M	8929	8929/M	8929/P
		8952	8952	8952/M	8952	8952/M	8952	8952/M	8952/P
lock	8849	8926	8926	8926/M	8926	8926/M	8926	8926/M	8926/M
		8926	8926	8926					
lock	9295	9338	9338	9338/M	9338	9338/M	9338	9338/M	9338/M
		9338	9338	9338					
lock	9295	9340	9340	9340/M	9340	9340/M	9340	9340/M	9340/P
lock	9507	9535	9535	9535/M	9535	9535/M	9535	9535/M	9535/M
		9535	9535	9535					
lock	9507	9538	9538	9538/M	9538	9538/M	9538	9538/M	9538/P
lock	9781	9817	9817	9817/M	9817	9817/M	9817	9817/M	9817/M
		9817	9817	9817					
lock	9781	9819	9819	9819/M	9819	9819/M	9819	9819/M	9819/P
lock	9985	10009	10009	10009/M	10009	10009/M	10009	10009/M	10009/P
		10019	10019	10019/M	10019	10019/M	10019	10019/M	10019/P
		10028	10028	10028/M	10028	10028/M	10028	10028/M	10028/P
lock_id	2142	5710	5746	5774	7516	7645	7914	7955	8022
		8094	8245	8520	8555	8599	8711	8896	8907
		8929	8952	9340	9538	9819	10009	10019	10028
lock_value	5695	5577/M	5621/M	5710/M	5746/M	5774/M	5797/M	7514/M	7516/M
		7612/M	7645/M	7712/M	7717/M	7814/M	7955/M	8022/M	8094/M
		8243/M	8245/M	8417/M	8520/M	8552/M	8555/M	8599/M	8711/M
		8896/M	8907/M	8926/M	8929/M	8952/M	9338/M	9340/M	9535/M
		9538/M	9817/M	9819/M	10009/M	10019/M	10028/M		
locked	2065	5577	5577	5606	5612	5710/M	5737/M	5797	5797
		7514	7514	7516/M	7612	7612	7645/M	7712	7712
		7717	7717	7914/M	7955/M	8022/M	8094/M	8243	8243
		8245/M	8417	8417	8520/M	8552	8552	8555/M	8599/M
		8711/M	8896/M	8907/M	8926	8926	8929/M	8952/M	9338
		9338	9340/M	9535	9535	9538/M	9817	9817	9819/M
		10009/M	10019/M	10028/M					
locked_fde_entry_p	7672	7712/P	7713/M	7714/M	7717/P	7718	7798/M	7803/M	7843/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7914/P	7921/M	7922/M	7929/P	7936/P	7938	7939	7955/P
locked_fde_p	8394	8417/P							
mapend	8735	8756/M	8759/M	8764	8773				
max_length	3845	7798	8448	8449/M	8451/M	8451	9818		
max_specified_transfer_size	7096	7874	7875						
media	2019	5417	7172	7615	7634	8553	9011	9546	
memory_check	9129	9129	9135	9137					
min_allocation_length	7575	7633	7635	7636					
mmc$	2926	2932	2935	2938	2941	2944	2947	2950	2954
		2957	2960	2963	2966	2969	2972	2976	2979
		2982	2985	2988	2991	2995	2998	3001	3004
		3007	3010	3013	3016	3019	3022	3025	3028
		3031	3034	3037	3040	3043	3046	3049	3052
		3055	3059	3062	3065	3068	3071	3074	3077
		3080	3083	3086	3089	3092	3095	3098	3101
		3104	3107	3110	3113	3116	3119	3123	3127
		3130	3134	3137	3140	3143	3146	3149	3152
		3155	3158	3161	3164	3167	3170	3173	3176
		3179	3182	3185	3188	3191	3195	3199	3202
		3205	3208	3211	3214	3217	3220	3223	3226
		3229	3232	3235	3238	3241	3244	3247	3250
		3253	3257	3260	3263				
mmc$as_random	3550	9705							
mmc$as_read_tu	3551	9708							
mmc$as_sequential	3550	9347	9702						
mmc$assign_active_escaped	4532	7607	7624						
mmc$assign_active_null	4531	4532	7652	8216	9543				
mmc$cell_pointer	3899	3904							
mmc$default_sdt_length	5328	5321	8160	8163					
mmc$eoi_uncertain	2119	5518	8446	8628					
mmc$failed_file_alloc_flag	5236	9100/P							
mmc$ha_binding	3879	7748	8466	9834					
mmc$ha_cache_bypass	3880	7761	8493	9847					
mmc$ha_execute	3879	7766	8481	9852					
mmc$ha_execute_global	3880	7772	8487	9858					
mmc$ha_execute_local	3880	7769	8484	9824	9855				
mmc$ha_read	3878	7743	8460	9829					
mmc$ha_read_key_lock	3878	8463							
mmc$ha_write	3879	7756	8472	9842					
mmc$ha_write_key_lock	3879	8475							
mmc$heap_pointer	3900	3908	7948						
mmc$kw_asid	3825	3861	7826						
mmc$kw_clear_space	3823	3848	7801	8504					
mmc$kw_current_segment_length	3822	3842	8445						
mmc$kw_error_exit_procedure	3824	3852	7794	8498	9820				
mmc$kw_gl_key	3824	3846	7799	8453					
mmc$kw_hardware_attributes	3826	3855	7742	8455	9823				
mmc$kw_inheritance	3826	3863	7844	8502					
mmc$kw_max_segment_length	3823	3844	7797	8447	9816				
mmc$kw_null_keyword	3821	7731	9812						
mmc$kw_preset_value	3825	3850	7802	8500					
mmc$kw_ps_transfer_size	3827	3871	7872						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmc$kw_ring_numbers	3821	3837	7732	8440	9813			
mmc$kw_segment_access_control	3825	3859	7804	8506	9867			
mmc$kw_segment_number	3822	3840	7740	8443				
mmc$kw_shadow_segment	3827	3865	7846					
mmc$kw_software_attributes	3824	3857	7781	8496				
mmc$kw_wired_segment	3827	3868	7840					
mmc$lss_none	4595	7110	8904	10016				
mmc$lus_free	4548	8905/P	10017/P					
mmc$pmo_user_stack	7026	7523/S						
mmc$pq_avail	6621	6667						
mmc$pq_free	6620	6679						
mmc$pq_job_fixed	6661	6668	6680					
mmc$pq_job_working_set	6663	6680	6681	7363				
mmc$pq_shared_first	6669	7366						
mmc$pq_shared_first_site	6671	6675						
mmc$pq_shared_last	6676	7366						
mmc$pq_shared_num_sites	6672	6675						
mmc$pq_shared_other	6630	6670						
mmc$pq_shared_site_01	6632	6671						
mmc$pq_shared_site_25	6656	6676						
mmc$pq_shared_task_service	6625	6669						
mmc$pq_swapped_io_error	6659	6679						
mmc$pq_wired	6623	6666						
mmc$ring_crossing_offset	7041	7523						
mmc$sa_fixed	3886	7601	7787					
mmc$sa_free_behind	3887	9349	9454	9457	9479	9704	9707	9711
mmc$sa_no_append	3888	7507	7543	9354	9729			
mmc$sa_read_transfer_unit	3887	7924	9349	9704	9707	9710	9714	
mmc$sa_stack	3887	7521	7536	7784	7787	8229	8247	8675
mmc$sa_wired	3886	7600	7787	7842				
mmc$sar_modify	4559	7507	7543	9354	9730			
mmc$sar_read	4559	7507	7543	9344	9354	9728		
mmc$sar_write_extend	4560	7110	7507	7543	8691	9354	9732	
mmc$sas_allow_access	4565	7109						
mmc$segment_fault_processor_id	5005	5059						
mmc$sequence_pointer	3899	3906	7945					
mmc$sf_get_segment_length_fde_p	3797	5520	8446	8628				
mmc$sf_set_segment_length_fde_p	3798	9772						
mmc$shadow_allocation_size	2923	7847	7851	8685				
mmc$si_new_segment	3918	8220	9946					
mmc$si_none	3917	7109	7980	7982	8160	8160	8202	8238	8895
mmc$si_share_segment	3917	7982	7983	8593	9049			
mmc$si_transfer_segment	3918	7980	7981	8237				
mmc$sr1_change_swap_file_queue	3732	3739						
mmc$sr1_delete_job_seg_by_sfid	3734	3740						
mmc$sr1_delete_seg_segnum	3721	3743	9014					
mmc$sr1_delete_seg_sfid	3722	3738	7175					
mmc$sr1_detach_file	3725	3738	8797					
mmc$sr1_end_job_recovery	3729	3749						
mmc$sr1_flush_avail_modified	3736	3741	9526					
mmc$sr1_flush_delete_seg_sfid	3726	3738	8797					
mmc$sr1_flush_seg_segnum	3727	3739	8797					
mmc$sr1_get_highest_offset	3733	3758						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmc$sr1_make_mfw_cache	3730	3752						
mmc$sr1_remove_detached_pages	3735	3740						
mmc$sr1_remove_job_shared_pages	3731	3754	7386					
mmc$sr1_replace_sfid	3728	3745						
mmc$srs_not_reserved	4605	7110	7892	9278/P	9320	9632		
mmc$srs_reserved	4605	9636						
mmc$srs_reserved_shared_stack	4606	7894	8160	8162	8204	8255	8258	9634
mmc$ssk_none	4615	4587	7110	7513	7864	8209	8211	8925	8943
		10038						
mmc$ssk_read_only_file	4615	8689						
mmc$ssk_read_write_file	4615	10007						
mmc$ssk_segment_number	4616	4585	7868	8210	8942			
mmc$ssr1_flush_delete_seg_sfid	3780	3770						
mmc$ssr1_free_delete_seg_sfid	3780	3770						
mmc$ssr1_move_modified_df_page	3781	3772	9251					
mmc$volume_unavailable_flag	5237	9181/P						
mme$address_not_0_mod_16384	3152	7852/P						
mme$asid_specified	3028	7828/P						
mme$binding_attribute_invalid	2979	7752/P	7821/P	9838/P	9888/P			
mme$caller_not_in_read_bracket	2995	8623/P						
mme$caller_not_in_write_bracket	2998	9763/P						
mme$execute_global_invalid	2982	7774/P	7813/P	9860/P	9880/P			
mme$execute_local_invalid	3001	9825/P	9869/P					
mme$illegal_segment_origin_chg	3188	7985/P						
mme$init_shadow_improper_seg	3158	8676/P						
mme$invalid_asid_specified	3031	7837/P						
mme$invalid_close_segment_req	2991	8897/P						
mme$invalid_ring_brackets	2976	7735/P						
mme$invalid_sfid	2932	7719/P						
mme$invalid_shadow_segment	3155	7865/P	10008/P					
mme$invalid_shared_taskid	3191	7701/P	8878/P					
mme$invalid_task_id	3052	8152/P	9937/P					
mme$io_write_error	3062	8795						
mme$length_not_0_mod_16384	3149	7848/P						
mme$no_write_access	3086	9766/P						
mme$ring_violation	2972	7976/P	8011/P	8547/P	9445/P	9699/P	9804/P	
mme$segment_not_pageable	3065	7602/P						
mme$segment_number_is_in_use	3010	7908/P						
mme$segment_number_not_in_use	3013	7899	8061/P	8407/P	8415/P	10087/P	10095/P	
mme$segment_number_too_big	3016	8055/P	8405/P	10085/P				
mme$segment_table_is_full	3007	7234/P						
mme$set_unmodifiable_attribute	3004	9894/P						
mme$software_attribute_invalid	2985	7789/P						
mme$unable_to_assign_fde	3260	7709/P	8234/P	8682/P				
mme$unsupported_keyword	3019	7795/P	8516/P	9821/P				
mmk$job_base	3470	3271	3275	3279	3283	3287	3291	3295	3299
		3303	3306	3310	3314	3317	3321	3324	3327
		3331	3335	3339	3343	3346	3349	3352	3355
		3358	3361	3364	3368	3372	3376	3380	3384
		3388	3392	3396	3400	3404	3408	3412	3416
		3420	3424	3428	3432	3436	3440		
mmp$advise_in	5839	9140						
mmp$asid	7148	7159	7355					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmp$assign_contiguous_memory	5853	7242	8177						
mmp$assign_mass_storage	7572	7658	9411	9537	9547				
mmp$asti	7129	7145							
mmp$build_segment	7662	7957							
mmp$change_seg_inheritance_r1	7960	7988							
mmp$change_segment_number_r1	7992	8024							
mmp$change_stack_attribute_r1	8027	8039							
mmp$close_asid_based_segment	8042	8071							
mmp$close_device_file	8074	8096							
mmp$convert_ps_transfer_size	5863	5888	7626	7877	7929	7931	9352	9460	
mmp$create_inherited_sdt	8121	8264							
mmp$delete_non_inherited_segs	8282	8305							
mmp$fetch_offset_mod_pages_r1	8308	8386							
mmp$fetch_sdt_sdtx_locked_fde	8390	8005	8086	8420	8437	8587	8664	8885	9999
mmp$fetch_segment_attributes_r1	8424	8522							
mmp$free_pages	5891	7241	7275	8176	9964				
mmp$get_allocated_addresses_r1	8526	8569							
mmp$get_max_sdt_sdtx_pointer	5899	5909	7306	8159	8297	8938	9046	9528	9941
mmp$get_sdt_entry_p	5913	7504	7540	8059	8249	8412	8765	8917	9001
		9284	9329	9629	10092				
mmp$get_sdt_entry_p	5915	5917/M	7504/M	7540/M	8059/M	8249/M	8412/M	8765/M	8917/M
		9001/M	9284/M	9329/M	9629/M	10092/M			
mmp$get_sdt_for_job_template	8573	8601							
mmp$get_sdtx_entry_p	5925	7505	7541	7596	8250	8259	8413	9003	9286
		9330	9630	10093					
mmp$get_sdtx_entry_p	5927	5929/M	7505/M	7541/M	7596/M	8250/M	8259/M	8413/M	9003/M
		9286/M	9330/M	9630/M	10093/M				
mmp$get_segment_length_r1	8604	8630	8694						
mmp$init_system_privilege_map	8729	8778							
mmp$initiate_shadowing_r1	8649	8713							
mmp$invalidate_segment	8849	8020	8301	8965					
mmp$issue_ring1_segment_request	8781	8819							
mmp$job_delete_inherited_sdt	8982	9030							
mmp$job_multiprocessing_control	9033	9060							
mmp$mfh_for_segment_manager	9072	9108							
mmp$mfh_shadow_file_reference	9111	9146							
mmp$mfh_volume_unavailable	9149	9187							
mmp$mm_move_mod_server_page	9238	9260							
mmp$open_asid_based_segment	9263	9291							
mmp$open_file_by_sfid	9295	9360							
mmp$os_preallocate_file_space	9378	9417							
mmp$preset_page_streaming_r1	9420	9486							
mmp$process_file_alloc	9507	9089	9096	9575					
mmp$reserve_segment_number_r1	9600	9641							
mmp$set_access_mode	9657	9676							
mmp$set_access_selections_r1	9679	9719							
mmp$set_segment_access_rights	9723	7507	7543	9354	9735				
mmp$set_segment_length_r1	9738	9777							
mmp$store_segment_attributes_r1	9781	9902							
mmp$task_delete_inherited_sdt	9920	9970							
mmp$terminate_shadowing_r1	9985	10043							
mmp$touch_all_pages	5938	5959	8338						
mmp$unlock_segment	5962	8905	10017						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmp$validate_segment_number	10071	7859	7898	7970	8346	8542	8615	9394	9437
		9691	9755	9798	10098				
mmt$access_selections	3550	9299	9682						
mmt$active_segment_table	6804	6989							
mmt$active_segment_table_entry	6789	6767	6805	6834	7343				
mmt$ast_index	2110	2006	3748	4442	6315	6782	7130	7148	7152
		7345							
mmt$attribute_descriptor	3835	3807	8427	9784					
mmt$attribute_keyword	3821	3836							
mmt$eoi_state	2119	2008							
mmt$global_page_queue_index	6679	6898							
mmt$global_page_queue_list_ent	6888	6898							
mmt$hardware_attribute_set	3890	3856	8456	8460	8463	8466	8472	8475	8481
		8484	8487	8493					
mmt$hardware_attributes	3878	3890							
mmt$job_page_queue_index	6680	6606	6899						
mmt$job_page_queue_list	6899	6270							
mmt$link	6812	6790	6824	6825	6885				
mmt$lock_segment_status	4595	4496							
mmt$locked_page	6846	6830							
mmt$lus_page_disposition	4547	5963							
mmt$max_sdt	4452	4456							
mmt$max_sdt_p	4456	5901	7296	8136	8287	8864	9038	9517	9929
mmt$max_sdtx	4520	4524							
mmt$max_sdtx_p	4524	5902	7297	8137	8288	8865	9039	9518	9930
mmt$memory_reserve_request	6866	6263							
mmt$page_age	6853	6833	6857	6857					
mmt$page_frame_index	6750	6742	6744	6745	6746	6814	6814	6868	6869
mmt$page_frame_queue_id	6681	6743	6798	6828					
mmt$page_frame_table_entry	6823	6765	6839						
mmt$page_map_offsets	7029	7018							
mmt$page_map_offsets_ord	7026	7029							
mmt$page_queue_list_entry	6884	6889	6899						
mmt$rb_change_segment_table	3557	7210							
mmt$rb_fetch_offset_mod_pages	3588	8324							
mmt$rb_ring1_segment_request	3715	7168	7346	8782	8861	8986	9515	9928	9993
mmt$rb_ring1_server_seg_request	3766	9245							
mmt$rb_set_get_segment_length	3790	5516	9751						
mmt$sdtx_stream_data	4503	4499							
mmt$segment_access_condition	5032	5060							
mmt$segment_access_rights	4559	4495	9300						
mmt$segment_access_state	4565	4490							
mmt$segment_attrib_descriptor	3802	7663							
mmt$segment_descriptor	4439	4449	4453	5915	7105	7105	7325	7327	7347
		7448	7458	7461	7464	7586	7675	7677	7684
		7967	8000	8002	8048	8081	8127	8141	8325
		8392	8433	8539	8575	8582	8612	8660	8737
		8862	8871	8988	9264	9270	9307	9308	9312
		9390	9433	9608	9686	9724	9752	9793	9994
		10073							
mmt$segment_descriptor_extended	4488	4517	4521	5927	5930	7109	7109	7276/P	7449
		7459	7462	7465	7505	7541	7587	7596	7676
		7678	7685	7968	8001	8003	8082	8128	8142

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8250	8259	8326	8393	8413	8434	8540	8576
		8583	8613	8661	8863	8989	9003	9265	9271
		9286	9309	9330	9391	9434	9609	9630	9687
		9725	9753	9794	9995	10074	10093		
mmt$segment_descriptor_table	4448	7208	8131						
mmt$segment_descriptor_table_ex	4516	7209	8132						
mmt$segment_inheritance	3917	3864	4492	7963					
mmt$segment_pointer	3902	7665	8286						
mmt$segment_pointer_kind	3899	3805	3903						
mmt$segment_reservation_state	4605	4493	7291	7681	9310				
mmt$server_segment_request	3780	3769							
mmt$set_get_subfunction_codes	3797	3794							
mmt$shadow_info	4580	4497							
mmt$shadow_reference_info	4619	4102							
mmt$shadow_segment_kind	4615	4584	8652						
mmt$software_attribute_set	3892	3858	4494	7536	7784	7785	7786	7788	7842
		8145	8247	9348	9457	9479	9704	9707	9710
		9711							
mmt$software_attributes	3886	3892							
mmt$xcb_page_wait_info	4630	4088							
mmv$a_divisor	6998	7138	7157	7157	7355	7355			
mmv$a_mult	6999	7138	7138	7157	7355				
mmv$ast_p	6989	7351	7358						
mmv$default_sdt_entry	7105	7690	9335						
mmv$default_sdtx_entry	7108	7691	9333						
mmv$first_transient_seg_index	7007	7304	9616						
mmv$page_map_offsets	7018	7523							
mmv$page_streaming_prestream	7047	7925	9350	9467	9468	9715	9716		
mmv$page_streaming_transfer	7052	7930	7931/P						
mmv$pf_job_template	9154	9178/M	9178						
mmv$pf_system_core	9153	9166/M	9166						
mmv$sfid_match	7101	7371/M	7371						
mmv$sfid_mismatch	7102	7373/M	7373						
mmv$sparse_threshold	7103	7633							
monitor_status	6113	6118	6120/P						
monitor_status	7187	7264	7264/P						
monitor_status	8308	8362	8362/P						
monitor_status	9238	9258	9258/P						
mp_p	8736	8744/M	8767/M	8774/M					
mtt$monitor_interlock	2126	1996							
multiprocessing_allowed	6262	7486							
nat$received_message_descriptor	4646	4639	4648						
nat$received_message_list	4638	4070							
new_fde_p	8657	8680/P	8681	8685/M	8686/M	8687/M	8701/M		
new_sdt_entry	8000	8015/M	8018/P						
new_sdt_length	3562	7262/M							
new_sdt_length	7201	7221/M	7223	7224/M	7224	7225	7226/M	7226	7229
		7230/M	7233	7238	7240	7252	7262		
new_sdt_length	8134	8170/M	8171						
new_sdt_offset	3560	7260/M							
new_sdt_p	7208	7238	7241/P	7242/P	7247/P	7247/P	7260		
new_sdtx_entry	8001	8016/M	8018/P						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
new_sdtx_offset	3561	7261/M							
new_sdtx_p	7209	7252	7254/P	7254/P	7261				
new_segment_number	7994	8018/P							
new_sfid	8658	8680/P	8702						
new_table_size	7202	7225/M	7241/P	7242/P					
new_table_size	8135	8171/M	8176/P	8177/P					
nlc$cc_connect_confirm	4678	4669							
nlc$cc_connect_request	4677	4667							
nlc$cc_expedited_data	4683	4669							
nlc$cc_max_pdu_kind	4685	4688							
nlc$channel_connection_pdu	4701	4653							
nlc$channelnet_pdu	4701	4655							
nlt$cc_pdu_kind	4688	4666							
nlt$cc_seq#_or_connect_time	4665	4654							
nlt$cc_sequence_number	4691	4670							
nlt$device_identifier	4698	4649							
nlt$pdu_type	4701	4652							
normal	1075	6118/M	6119	7212/M	7243	7264/M	7264	7265	7303/M
		7312	7592/M	7631	7650	7689/M	7861	7899	7900/M
		7903	7907	7913	7971	8006	8035/M	8051/M	8084/M
		8148/M	8178	8292/M	8329/M	8347	8362/M	8362	8363
		8400/M	8436/M	8543	8588	8616	8665	8813	8886
		8906	8994/M	9043/M	9090	9097	9098	9101	9182
		9247/M	9258/M	9258	9275/M	9279	9315/M	9322	9395
		9400	9412	9415	9436/M	9438	9522/M	9548	9615/M
		9624	9692	9756	9799	9934/M	10000	10018	10080/M
normal	3570	6118	7264	8362	8794	9258			
offset	5437	5422/M	8553/M						
offset	8730	8745							
offset_array	8317	8323							
offset_list	3593	8358/M							
offset_list	8312	8333	8357	8370	8380/M				
offset_p	8323	8333	8338/P	8338/P	8358	8364	8372	8380	8384
offsets_returned	3592	8357/M	8370	8371	8373	8379	8383		
offsets_returned	8313	8373/M	8383/M						
old_sdt_length	7203	7219/M	7274	7276/P					
old_sdt_offset	7204	7217/M	7278	7279					
old_sdt_p	7206	7279/M	7281						
old_sdt_p	8002	8005/P	8015						
old_sdtx_offset	7205	7218/M	7275/P	7280					
old_sdtx_p	7207	7280/M	7282						
old_sdtx_p	8003	8005/P	8010	8016					
old_segment_number	7993	8005/P	8020/P						
open_count	2002	7511/M	7511	7515/M	7515	7545/M	7545	8093/M	8093
		8224/M	8244/M	8244	8686/M	8919/M	8919	8927/M	8927
		8928	8950	8951/M	9019/M	9339/M	9339	9950	9953/M
		9953	10026/M	10026	10027				
open_count	8859	8924/M	8945/M	8945	8950/M	8950	8951	8953	8957
open_count	9991	10027/M	10030						
open_validating_ring_number	4489	7694/M	8010	8242	8300	8594/M	8674	9004	9050
		9803							
osc$aging_interval_maximum	6935	6938							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$asid_ei	3942	7832							
osc$asid_eie	3943	7833							
osc$asid_nos	3944	7834							
osc$base_exception	2217	2223							
osc$binding_segment	3933	7750	7819	8464	9836	9886			
osc$call_instruction	4896	4904							
osc$data_read	4895	4904							
osc$free_running_clock_maximum	1723	1720							
osc$global_privilege	3930	7811	8485	9878					
osc$invalid_ring	534	574							
osc$local_privilege	3930	7770	8482	9856	9868				
osc$max_fault_contents	5072	5066							
osc$max_name_size	811	815	818	1022					
osc$max_number_of_processors	7074	7069							
osc$max_page_frames	6686	6310	6311	6605	6607	6750	6791	6886	6892
osc$max_page_size	1503	1499							
osc$max_page_table_entries	6687	6690							
osc$max_ring	533	574	575						
osc$max_segment_length	557	580	763	4500	4531	7111			
osc$max_status_condition_code	1090	1086	1102						
osc$max_string_size	1106	1109	1112	1117	5985				
osc$max_tasks	2083	2080							
osc$maximum_offset	556	557	577	577	578	7938	7941		
osc$maximum_processor_id	4944	4940							
osc$maximum_processors	7078	7074							
osc$maximum_segment	555	576							
osc$min_ecc	2216	2217							
osc$min_page_size	1502	1499							
osc$min_ring	532	575							
osc$non_executable	3929	7106	7478	7478	7777	8767	9663	9863	
osc$non_privileged	3929	7767	8479	9853					
osc$non_readable	3932	7746	8546	9667	9832				
osc$non_writable	3935	7479	7507	7543	7759	9345	9354	9671	9727
		9765	9845						
osc$nowait	5382	7241/P	7638/P	8176/P	8905/P	10017/P			
osc$pr_base_constant	5646	5577	5603	5710	5725	5797	6024	6049	7472
		7514	7516	7594	7612	7645	7712	7717	7888
		7914	7955	8022	8036	8052	8094	8156	8243
		8245	8293	8401	8417	8520	8552	8555	8599
		8711	8749	8896	8907	8912	8926	8929	8937
		8952	8996	9044	9083	9099	9126	9180	9276
		9316	9338	9340	9527	9535	9538	9613	9817
		9819	10009	10019	10028	10081			
osc$pr_page_size_mask	5649	5948	8338						
osc$pva_purge_all_page_seg_map	3957	8068	8091	8240	8707	8962	9900	10041	
osc$pva_purge_segment_cache	3952	7552							
osc$read_key_lock_controlled	3932	8461							
osc$read_uncontrolled	3933	7106	7744	8458	9830				
osc$rtr_sft_full	4952	9563/P							
osc$segnum_job_fixed_heap	5297	5577	5602	5710	5724	5797	6023	6048	7275/P
		7279	7280	7472	7514	7516	7594	7612	7645
		7712	7717	7888	7914	7955	8022	8036	8052
		8094	8156	8243	8245	8293	8401	8417	8520

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8552	8555	8599	8711	8749	8896	8907	8912
		8926	8929	8937	8952	8996	9004	9044	9048
		9083	9099	9126	9180	9276	9316	9338	9340
		9527	9535	9538	9613	9817	9819	9961	9962
		10009	10019	10028	10081				
osc$task_time_slice_maximum	4274	4277							
osc$vl_cache_bypass	4467	7488	7493	7762	7806	8491	8507	9052	9342
		9848	9873						
osc$vl_invalid_entry	4467	7317	8060	8069	8092	8160	8161	8201	8241
		8256	8299	8414	8767	8918	8940	8963	9002
		9021	9048	9542	9631	9945	10094		
osc$vl_regular_segment	4467	7106	7764	7808	9054	9850	9875		
osc$wait	5382	7276/P	9965/P						
osc$write_key_lock_controlled	3935	8473							
osc$write_uncontrolled	3936	7106	7757	8470	8690	9843			
osk$base	3475	3964	3968	3972	3976	3980	3984	3988	3992
		3996	4000	4004	4008	4012	4016	4021	4024
		4027							
osk$system_class	3521	3505	3506	3507	3508	3509	3510	3511	
osp$begin_system_activity	5978	9523							
osp$end_system_activity	5980	9566	9574						
osp$mfh_for_segment_manager	5752	5710	5742	7516	7645	7914	7955	8022	8094
		8245	8520	8555	8599	8711	8896	8907	8929
		8952	9340	9538	9819	10009	10019	10028	
osp$set_status_abnormal	5982	6120	7234	7264	7602	7701	7709	7719	7735
		7752	7774	7789	7795	7813	7821	7828	7837
		7848	7852	7865	7908	7976	7985	8011	8055
		8061	8152	8234	8362	8405	8407	8415	8516
		8547	8623	8676	8682	8878	8897	9258	9445
		9699	9763	9766	9804	9821	9825	9838	9860
		9869	9880	9888	9894	9937	10008	10085	10087
		10095							
osp$system_error	5683	5710	5729	7266	7516	7645	7914	7955	8022
		8094	8245	8520	8555	8599	8711	8896	8907
		8908	8929	8952	8954	9102	9183	9340	9538
		9819	9951	10009	10019	10020	10028		
ost$aging_interval	6938	6196	6197						
ost$asid	1712	453	1660	1708	3862	4473	6254	6769	6780
		6781	6796	7129	7133	7149	7344		
ost$binary_unique_name	1411	1403	1514	1625	1998				
ost$byte_count	1702	5840	5847	5892	5971	6103			
ost$caller_identifier	4038	8126	8856						
ost$clear_file_space	1016	463	618	2886					
ost$cp_time	4734	4087	6487	6517					
ost$cp_time_value	4732	4100	4735	4736	6191	6192	6296		
ost$cs_lock	2140	4068							
ost$debug_code	4895	4883							
ost$debug_list	4891	4795							
ost$debug_list_entry	4882	4891							
ost$debug_mask	4901	4794							
ost$exchange_package	4744	4055							
ost$execute_privilege	3929	3924	4468						
ost$execution_control_block	4054	4080	5600	5722	5900	5914	5926	6011	6029

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		6047	6976	7188	7290	7466	7467	7589	7686
		7687	8033	8049	8129	8138	8290	8398	8739
		8870	8872	8992	9041	9079	9124	9159	9273
		9313	9520	9610	9932	10078			
ost$family_name	6948	6943							
ost$flags	4801	4751							
ost$frame_descriptor	4859	4874							
ost$free_running_clock	1720	2014	4086	4226	6190	6199	6265	6266	6267
		6268	6302	6312	6313	6314	6413	6795	
ost$global_task_id	2074	2016	2067	4065	4066	4956	5159	5599	5721
		6034	6055	6186	6259	6288	9077	9152	
ost$heap	5314	7063							
ost$key_lock	563	3847	4474						
ost$key_lock_value	569	566	4040	4042	4818	4820			
ost$keypoint_class	4833	4764	4835						
ost$keypoint_mask	4835	4767							
ost$minimum_save_area	4869	4756	4844	5053	9155				
ost$monitor_condition	4705	4712							
ost$monitor_conditions	4712	4757	4761	4849	5128	5142			
ost$monitor_fault	5049	4998							
ost$monitor_fault_contents	5066	5062							
ost$name	818	781	842	888	919	1292	1624	1633	4189
		4405	5091	6108	6907	6946	6948		
ost$p_register	4816	4745	4870	5120	5126				
ost$page_id	6692	6702							
ost$page_size	1499	1480	5944	7087					
ost$page_table_entry	6697	6706	6766						
ost$page_table_index	6690	6706	6831						
ost$paging_statistics	4922	4095	6518						
ost$processor_id	4940	4058	4934						
ost$processor_id_set	4934	4057							
ost$processor_model_number	1429	1413	6954						
ost$processor_serial_number	1507	1412	6955						
ost$pva	585	4789	4807	4821	5050	5143	7460		
ost$read_privilege	3932	3925	4469						
ost$register_number	4812	4786	4855	4863	4864	4865			
ost$ring	574	586	3838	3839	4043	4471	4472	4489	4806
ost$ring1_termination_reason	4952	4091	6129						
ost$segment	576	587	2015	3562	3744	3756	3841	4044	4586
		4784	4885	5497	5915	5927	6981	7007	7203
		7292	7329	7452	7573	7680	7961	7993	7994
		8043	8075	8143	8146	8289	8309	8391	8425
		8527	8584	8605	8733	8735	8850	8866	8987
		8990	9040	9266	9272	9301	9311	9389	9421
		9602	9611	9680	9739	9782	9789	9931	9986
		10072							
ost$segment_access_control	3922	3860							
ost$segment_descriptor	4466	4440	8738	9658					
ost$segment_length	580	505	2906	3793	3843	3845	3867	3869	3872
		5353	5459	5855	5940	7202	7575	7583	7679
		8135	8607	8659	9380	9741			
ost$segment_offset	577	588	1709	3593	3774	4504	4886	4888	5448
		5458	8312	8317	8529	8730	9241		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$signature_lock	2141	434	5755						
ost$stack_frame_save_area	4843	4877	5086						
ost$status	1074	1059	1286	3854	5121	5354	5366	5390	5398
		5404	5452	5476	5482	5489	5684	5841	5848
		5856	5894	5965	5973	5986	5996	6004	6056
		6115	7165	7189	7293	7576	7666	7964	7996
		8030	8044	8076	8123	8133	8283	8314	8395
		8428	8533	8577	8608	8653	8786	8853	8991
		9035	9078	9123	9156	9242	9267	9302	9381
		9386	9427	9509	9519	9603	9683	9742	9785
		9922	9987	10075					
ost$status_condition	1098	3571							
ost$status_condition_code	1102	1077	1098	5984					
ost$status_identifier	5991	5983							
ost$string	1115	1078							
ost$string_size	1109	1116							
ost$system_flag	5227	5223	6054						
ost$system_privilege_map	5321	8736	8754	8759	8773				
ost$system_virtual_address	1707	6836							
ost$task_index	2080	2075	2103	2104					
ost$task_time_slice	4277	4263							
ost$top_of_stack_pointer	4804	4796							
ost$trap_enable	4838	4753	5117						
ost$user_condition	4715	4722							
ost$user_conditions	4722	4755	4759	4847	4876	5089	5129		
ost$user_identification	6941	6185							
ost$user_name	6946	6942							
ost$valid_relative_pointer	583	2021	2024	4084	4085				
ost$valid_ring	575	981	982	983	3803	4796	7962	7995	8029
		8426	8528	8606	8651	8851	9297	9298	9422
		9681	9740	9783					
ost$virtual_machine_identifier	4826	4747	4749	4871					
ost$wait	5382	5364	5893	5964	5972				
ost$write_privilege	3935	3926	4470						
ost$x_register	4813	4786	4855						
osv$cpus_logically_on	7069	7550							
osv$job_fixed_heap	7063	7238	7252	7278	7281	7282	8167	8188	8333
		8364	8372	8384	9967	9968			
osv$multiple_cpus_possible	7082	7477	9341						
osv$page_size	7087	5876	5876	7223	7224	7224	7225	7226	7240
		7274	7523	7626	7626	7856	7858	7877	7877
		7929	7929	7931	7931	8169	8170	8170	8171
		8175	8700	9132/P	9141/P	9144/P	9352	9352	9460
		9460	9963						
P	5433	5421/M	5423	8553/M	8553				
P	5572	5577							
P	5595	5623							
P	5706	5710	5710						
P	5706	5710							
P	5715	5746	5746						
P	5715	5746							
P	5755	5773	5774						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
p	5765	5772							
p	5789	5797							
p	7447	7514							
p	7447	7516	7516						
p	7447	7516							
p	7572	7612							
p	7572	7645	7645						
p	7572	7645							
p	7662	7712	7717						
p	7662	7914	7914	7955	7955				
p	7662	7914	7955						
p	7992	8022	8022						
p	7992	8022							
p	8074	8094	8094						
p	8074	8094							
p	8121	8243							
p	8121	8245	8245						
p	8121	8245							
p	8390	8417							
p	8424	8520	8520						
p	8424	8520							
p	8526	8552							
p	8526	8555	8555						
p	8526	8555							
p	8573	8599	8599						
p	8573	8599							
p	8649	8711	8711						
p	8649	8711							
p	8849	8896	8896	8907	8907	8929	8929	8952	8952
p	8849	8896	8907	8929	8952				
p	8849	8926							
p	9295	9338							
p	9295	9340	9340						
p	9295	9340							
p	9507	9535							
p	9507	9538	9538						
p	9507	9538							
p	9781	9817							
p	9781	9819	9819						
p	9781	9819							
p	9985	10009	10009	10019	10019	10028	10028		
p	9985	10009	10019	10028					
p1	5693	5577/M	5623/M	5710/M	5746/M	5772/M	5797/M	7514/M	7516/M
		7612/M	7645/M	7712/M	7717/M	7914/M	7955/M	8022/M	8094/M
		8243/M	8245/M	8417/M	8520/M	8552/M	8555/M	8599/M	8711/M
		8896/M	8907/M	8926/M	8929/M	8952/M	9338/M	9340/M	9535/M
		9538/M	9817/M	9819/M	10009/M	10019/M	10028/M		
p2	5694	5577/M	5624/M	5710/M	5746/M	5773/M	5797/M	7514/M	7516/M
		7612/M	7645/M	7712/M	7717/M	7914/M	7955/M	8022/M	8094/M
		8243/M	8245/M	8417/M	8520/M	8552/M	8555/M	8599/M	8711/M
		8896/M	8907/M	8926/M	8929/M	8952/M	9338/M	9340/M	9535/M
		9538/M	9817/M	9819/M	10009/M	10019/M	10028/M		
p_register	2068	5577/M	5615/M	5797/M	7514/M	7612/M	7712/M	7717/M	8243/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8417/M	8552/M	8926/M	9338/M	9535/M	9817/M		
p_register_2	2069	5577/M	5608/M	5797/M	7514/M	7612/M	7712/M	7717/M	8243/M
		8417/M	8552/M	8926/M	9338/M	9535/M	9817/M		
page_count	4622	9130	9141/P	9144/P					
page_size	5944	5948/M	5954						
page_size	8308	8338/M	8338						
page_streaming_transfer_size	7674	7873/M	7874	7875/M	7877/P				
page_streaming_ts_shift	7585	7626/P	7627						
page_streaming_ts_shift	7673	7877/P	7878	7929/P	7931/P	7933			
page_streaming_ts_shift	9306	9352/P	9353						
page_streaming_ts_shift	9432	9460/P	9462	9463					
pages_to_touch	5939	5949							
pages_to_touch	8308	8338							
parent_sdt_p	8136	8159/P	8160	8161	8201	8205	8241/M	8256	
parent_sdtx_p	8137	8159/P	8160	8160	8160	8162	8202	8203	8208
		8237	8254	8257					
parent_xcb_p	8138	8156/P	8158	8159/P					
passive_for_shadow_by_segnum	4588	7871/M	8212/M	8936	8944/M	10039/M			
pfc$append	1024	9672							
pfc$execute	1025	1027	1030	9664					
pfc$modify	1024	9673							
pfc$read	1024	1027	1030	9668					
pfc$shorten	1024	9672							
pft$share_options	1030	1031							
pft$usage_options	1027	1028							
pft$usage_selections	1028	612	9659	9661	9664	9668	9672	9672	9673
		9788							
pmc$internal_base_exception	6075	6079	6086	6095					
pmc$kill_task_flag	5227	5243							
pmc$max_signal_contents	5210	5204							
pmc$max_task_id	4965	4962							
pmc$min_ecc	6067	6063							
pmc$pc_base_exception	6063	6075							
pmp$delay	6002	9095	9173	9413					
pmp$find_executing_task_xcb	6010	6025	7472	7594	7888	8036	8052	8156	8293
		8401	8749	8912	8937	8996	9044	9083	9126
		9276	9316	9527	9613	10081			
pmp$find_task_xcb	6028	7538	7699	8150	8876	8915	9935		
pmp$get_executing_task_gtid	6033	6051	9099	9180					
pmp$set_system_flag	6054	9100	9181						
pmp$zero_out_table	6101	7247	7254	8182	8189				
pmt$binary_mainframe_id	6953	6189							
pmt$condition_identifier	5038	5033							
pmt$cpu_model_number	1489	1478	1485						
pmt$cpu_serial_number	1492	1479	1484						
pmt$initialization_value	2184	2013	3851	7055					
pmt$program_name	919	620	624	640	689	907			
pmt$sense_switches	6962	6200							
pmt$signal	5166	5160							
pmt$signal_contents	5204	5168							
pmt$signal_id	5171	5167							
pmt$task_id	4962	4082	4957	6028	7451	7664	8122	8852	9921
pmt$task_private_descriptor	6980	6977							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pmt$task_template	6975	6971							
pointer_kind	3805	7944							
power	5868	5877/M	5879/M	5879	5882	5885			
power	7572	7626/M	7626/M	7626	7626	7626			
power	7662	7877/M	7877/M	7877	7877	7877	7929/M	7929/M	7929
		7929	7929	7931/M	7931/M	7931	7931	7931	
power	9295	9352/M	9352/M	9352	9352	9352			
power	9420	9460/M	9460/M	9460	9460	9460			
preset_and_save_fb_and_ts	9423	9449							
preset_streaming	4510	9466/M	9476/M						
preset_value	2013	7803/M	8501						
preset_value	3851	7803	8501/M						
process_virtual_address	9379	9393							
ps_transfer_size	3872	7873							
ps_transfer_size	5863	5876							
ps_transfer_size	7572	7626							
ps_transfer_size	7662	7877	7929	7931					
ps_transfer_size	9295	9352							
ps_transfer_size	9420	9460							
ps_transfer_size_power	5864	5882	5883/M	5883	5885/M				
ps_transfer_size_power	7572	7626	7626/M	7626	7626/M				
ps_transfer_size_power	7662	7877	7877/M	7877	7877/M	7929	7929/M	7929	7929/M
		7931	7931/M	7931	7931/M				
ps_transfer_size_power	9295	9352	9352/M	9352	9352/M				
ps_transfer_size_power	9420	9460	9460/M	9460	9460/M				
psa	9155	9161/M	9164						
pva	4807	7525/M	8997						
pva	5415	5421/M	5422/M	5423					
pva	5435	5422/M	8553/M						
pva	8047	8067/M	8068						
pva	8080	8090/M	8091						
pva	8139	8239/M	8240						
pva	8526	8553/M	8553/M	8553					
pva	8574	8586							
pva	8860	8890/M	8905/P	8962					
pva	9792	9899/M	9900						
pva	9992	9998/M	10017/P	10041					
pva_p	7460	7524/M	7525						
queue_id	6798	7363	7366	7366					
queue_status	2012	7364	7365	7368	7369	7384	7485	7491	7713/M
r1	3838	7733	7738	8441/M	9814				
r1	4471	7522	7525/S	7526	7692/M	7738/M	7946/P	7949/P	7952
		8230	8441	8674	9336/M	9762	9814/M		
r1	9297	9336							
r2	3839	7733	7734	7739	8442/M	9815			
r2	4472	7693/M	7739/M	7975	8442	8546	8622	9337/M	9444
		9444	9698	9815/M					
r2	9298	9337							
rb	7168	7174/M	7175/M	7176/M	7177/P	7177/P			
rb	7346	7385/M	7386/M	7387/M	7388/M	7389/M	7390/P	7390/P	
rb	8782	8791/M	8793/P	8793/P	8794	8795	8796	8799	8816/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
rb	8986	9013/M	9014/M	9015/M	9016/P	9016/P			
rb	9515	9525/M	9526/M	9544/M	9545/P	9550/P	9550/P	9557/P	
rb_ring1_server_seg_request	9245	9249/M	9250/M	9251/M	9252/M	9253/M	9255/P	9255/P	9257
		9258/P							
read_privilege	3925	7819	7824	8513/M	9886	9891			
read_write_access_selection	9300	9344							
referenced_byte	5945	5953/M	5957/M						
referenced_byte	8308	8338/M	8338/M						
reqcode	3589	8355/M							
reqcode	3716	7174/M	7385/M	9013/M	9525/M				
reqcode	3767	9249/M							
reqcode	5691	5577/M	5619/M	5710/M	5746/M	5769/M	5797/M	7514/M	7516/M
		7612/M	7645/M	7712/M	7717/M	7914/M	7955/M	8022/M	8094/M
		8243/M	8245/M	8417/M	8520/M	8552/M	8555/M	8599/M	8711/M
		8896/M	8907/M	8926/M	8929/M	8952/M	9338/M	9340/M	9535/M
		9538/M	9817/M	9819/M	10009/M	10019/M	10028/M		
request	3721	7175/M	7386/M	8796	9014/M	9526/M			
request	3769	9251/M							
request_block	5516	5519/M	5520/M	5521/M	5522/P	5522/P			
request_block	7210	7259/M	7260/M	7261/M	7262/M	7263/P	7263/P	7264/P	
request_block	8324	8355/M	8356/M	8357/M	8358/M	8359/M	8361/P	8361/P	8362/P
		8370	8371	8373	8379	8383			
request_block	8424	8446/M	8446/M	8446/M	8446/P	8446/P			
request_block	8604	8628/M	8628/M	8628/M	8628/P	8628/P			
request_block	9751	9770/P	9771/M	9772/M	9773/M	9775/P	9775/P		
request_code	3558	7259/M							
request_code	3791	5519/M	8446/M	8628/M	9771/M				
residence	1948	5542	5543	5795	5795	7361	7362	7514	7514
		7612	7612	7712	7712	7717	7717	8243	8243
		8417	8417	8552	8552	8627	8627	8926	8926
		9006	9006	9338	9338	9535	9535	9545	9545
		9557	9557	9770	9770	9817	9817	9949	9949
return_unallocated_offsets	3594	8359/M							
return_unallocated_offsets	8311	8359							
ring	4043	8890							
ring_1_stack_segnum	8987	8997/M	9005						
rma	8140	8190/P	8191	8192					
rma	9122	9132/P	9133						
rmc$external_vsn_size	1592	1598							
rmc$msc_system_critical_files	5345	7619							
rmc$msc_user_temporary_files	5344	7621							
rmc$recorded_vsn_size	1595	1605							
rmc$unspecified_file_class	1380	1373							
rmt$external_vsn	1598	1612							
rmt$recorded_vsn	1605	495	1546	1558	1611	1846			
rmt$volume_descriptor	1610	1617							
rp	4469	7744/M	7746/M	7750/M	7824/M	8457	8513	8546	9667
		9830/M	9832/M	9836/M	9891/M				
saved_free_behind	9426	9454/M	9455	9477					
saved_transfer_size	9425	9459/M	9481	9482					
scan_sdt_for_inherited_segs	8999	8999	9026						
scan_sdt_for_inherited_segs	9943	9943	9957						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
sd	7447	7507	7543					
sd	9295	9354						
sd	9724	9727						
sd_p	9390	9394/P						
sd_p	9686	9691/P	9698					
sdt_entry	7347	7349/M	7351	7352	7354/M	7356/M	7357	7358/S
sdt_entry	7448	7469						
sdt_entry	7675	7690/M	7692/M	7693/M	7738/M	7739/M	7744/M	7746/M
		7757/M	7759/M	7762/M	7764/M	7767/M	7770/M	7777/M
		7806/M	7808/M	7816/M	7824/M	7825/M	7835/M	7936/P
		7949/P	7952					
sdt_entry	8141	8205/M	8206/M	8207/M	8230	8251		
sdt_entry	8575	8597/M	8598/M					
sdt_entry	9264	9285						
sdt_entry	9307	9335/M	9336/M	9337/M	9342/M	9345/M	9354/P	9356/P
sdt_entry_p	7461	7504/M	7510/P					
sdt_entry_p	7677	7898/P						
sdt_entry_p	7967	7970/P	7975					
sdt_entry_p	8048	8059/M	8060	8069/M				
sdt_entry_p	8081	8086/P	8092/M					
sdt_entry_p	8325	8346/P						
sdt_entry_p	8392	8412/M	8414					
sdt_entry_p	8433	8437/P	8441	8442	8454	8457	8469	8478
		8507	8512	8513	8514			
sdt_entry_p	8538	8542/P	8546	8546				
sdt_entry_p	8582	8587/P	8597					
sdt_entry_p	8612	8615/P	8622					
sdt_entry_p	8660	8664/P	8674	8690/M	8706/M			
sdt_entry_p	8737	8765/M	8766					
sdt_entry_p	8862	8885/P	8963/M					
sdt_entry_p	8988	9001/M	9002	9021/M				
sdt_entry_p	9270	9284/M	9285/M					
sdt_entry_p	9308	9329/M	9356/P					
sdt_entry_p	9433	9437/P	9444	9444				
sdt_entry_p	9608	9629/M	9631					
sdt_entry_p	9752	9755/P	9762	9765				
sdt_entry_p	9793	9798/P	9814/M	9815/M	9830/M	9832/M	9836/M	9843/M
		9848/M	9850/M	9853/M	9856/M	9863/M	9873/M	9875/M
		9891/M	9892/M					
sdt_entry_p	9994	9999/P	10036/M					
sdt_entry_p	10073	10092/M	10094					
sdt_offset	4084	5906	5918	7217	7306	7504	7540	8059
		8194/M	8249	8297	8412	8765	8917	8938
		9046	9284	9329	9528	9629	9941	9961
sdt_p	5901	5906/M						
sdt_p	7289	7306/M						
sdt_p	7296	7306/P	7317					
sdt_p	8121	8159/M						
sdt_p	8282	8297/M						
sdt_p	8287	8297/P	8299					
sdt_p	8849	8938/M						
sdt_p	8864	8938/P	8940					
sdt_p	9033	9046/M						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
sdt_p	9038	9046/P	9048	9052/M	9054/M			
sdt_p	9507	9528/M						
sdt_p	9517	9528/P	9542					
sdt_p	9920	9941/M						
sdt_p	9929	9941/P	9945					
sdtx	7447	7507/M	7507	7507/M	7507/M	7543/M	7543	7543/M
sdtx	9295	9354/M	9354	9354/M	9354/M			
sdtx	9725	9728/M	9729	9730/M	9732/M			
sdtx_entry	7449	7470	7521					
sdtx_entry	7676	7691/M	7694/M	7695/M	7724/M	7783/M	7792/M	7841/M
		7845/M	7855/M	7857/M	7868/M	7869/M	7870/M	7878/M
		7924	7925/M	7928	7933/M	7936/P		
sdtx_entry	8142	8208/M	8209	8210	8211/M	8212/M	8213/M	8213
		8220	8229	8232/M	8238/M	8242	8243/P	8246/M
		8252	8257/M	8258/M	8260			
sdtx_entry	8576	8596/M						
sdtx_entry	9265	9287						
sdtx_entry_p	7462	7505/M	7506	7509/M				
sdtx_entry_p	7587	7596/M	7597	7598	7599	7600	7601	7625
		7652/M						
sdtx_entry_p	7678	7898/P						
sdtx_entry_p	7968	7970/P	7980	7981/M	7982	7983/M		
sdtx_entry_p	8082	8086/P						
sdtx_entry_p	8326	8346/P	8350					
sdtx_entry_p	8393	8413/M	8417/P					
sdtx_entry_p	8434	8437/P	8497	8503				
sdtx_entry_p	8540	8542/P	8552/P	8566/P				
sdtx_entry_p	8583	8587/P	8593/M	8594/M	8595/M	8596		
sdtx_entry_p	8613	8615/P	8627/P					
sdtx_entry_p	8661	8664/P	8674	8675	8691/M	8696/M	8697/M	8697
		8699/M	8702/M					
sdtx_entry_p	8863	8885/P	8895	8904	8925	8926/P	8933/P	8936
		8959/P						
sdtx_entry_p	8989	9003/M	9004	9006/P	9020/P	9020/P		
sdtx_entry_p	9271	9286/M	9287/M					
sdtx_entry_p	9309	9330/M	9333/M	9334/M	9348/M	9350/M	9353/M	9354/P
sdtx_entry_p	9434	9437/P	9454	9456/M	9456	9459	9462	9463/M
		9466/M	9467	9468/M	9476/M	9478/M	9478	9481
sdtx_entry_p	9609	9630/M	9632	9634/M	9636/M			
sdtx_entry_p	9753	9755/P	9770/P					
sdtx_entry_p	9794	9798/P	9803	9817/P				
sdtx_entry_p	9995	9999/P	10007	10016	10031/P	10031/P	10037/M	10037
		10039/M						
sdtx_entry_p	10074	10093/M						
sdtx_offset	4085	5907	5930	7218	7306	7505	7541	7596
		8195/M	8250	8259	8297	8413	8938	9003
		9286	9330	9528	9630	9941	9962	10093
sdtx_p	5902	5907/M						
sdtx_p	7289	7306/M						
sdtx_p	7297	7306/P	7318					
sdtx_p	8121	8159/M						
sdtx_p	8282	8297/M						
sdtx_p	8288	8297/P	8300					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
sdtx_p	8849	8938/M							
sdtx_p	8865	8938/P	8941	8942	8943/M	8944/M			
sdtx_p	9033	9046/M							
sdtx_p	9039	9046/P	9049	9050					
sdtx_p	9391	9394/P	9399/P	9411/P					
sdtx_p	9507	9528/M							
sdtx_p	9518	9528/P	9543	9544					
sdtx_p	9687	9691/P	9703/M	9703	9706/M	9706	9709/M	9709	9714
		9715	9716/M						
sdtx_p	9920	9941/M							
sdtx_p	9930	9941/P	9946	9949/P	9954/P	9955/P			
sdtx_table	4521	7318	8160	8160	8160	8162	8202	8203	8208
		8237	8254	8257	8300	8941	8942	8943/M	8944/M
		9049	9050	9543	9544	9946	9949/P	9954/P	9955/P
seg	587	8997							
seg	5539	5541/M	5544/M	5546/M	5549	5550			
seg	5789	5795/M	5795/M	5795/M	5795	5795			
seg	7447	7514/M	7514/M	7514/M	7514	7514			
seg	7572	7612/M	7612/M	7612/M	7612	7612			
seg	7662	7712/M	7712/M	7712/M	7712	7712	7717/M	7717/M	7717/M
		7717	7717						
seg	8121	8243/M	8243/M	8243/M	8243	8243			
seg	8390	8417/M	8417/M	8417/M	8417	8417			
seg	8526	8552/M	8552/M	8552/M	8552	8552			
seg	8604	8627/M	8627/M	8627/M	8627	8627			
seg	8849	8926/M	8926/M	8926/M	8926	8926			
seg	8982	9006/M	9006/M	9006/M	9006	9006			
seg	9295	9338/M	9338/M	9338/M	9338	9338			
seg	9507	9535/M	9535/M	9535/M	9535	9535	9545/M	9545/M	9545/M
		9545	9545	9557/M	9557/M	9557/M	9557	9557	
seg	9738	9770/M	9770/M	9770/M	9770	9770			
seg	9781	9817/M	9817/M	9817/M	9817	9817			
seg	9920	9949/M	9949/M	9949/M	9949	9949			
seg_attributes	8427	8438	8438	8439	8441/M	8442/M	8444/M	8446/M	8448
		8449/M	8451/M	8451	8454/M	8456/M	8459/M	8459	8462/M
		8462	8465/M	8465	8471/M	8471	8474/M	8474	8480/M
		8480	8483/M	8483	8486/M	8486	8492/M	8492	8497/M
		8499/M	8501/M	8503/M	8505/M	8508/M	8510/M	8512/M	8513/M
		8514/M							
segment	6977	8744	8745						
segment_attributes	9784	9810	9810	9811	9814	9815	9818	9824	9829
		9834	9842	9847	9852	9855	9858	9868	9872
		9878	9883	9886	9891	9892			
segment_descriptor	9658	9663	9667	9671					
segment_inheritance	7963	7980	7982						
segment_length	3793	9773/M							
segment_length	7679	7939/M	7941/M	7946/P	7949/P				
segment_length	8607	8628/M							
segment_length	8659	8694/P	8699						
segment_length	9741	9773							
segment_lock	4496	8904	10016						
segment_num_list	9602	9619	9619	9638/M					
segment_number	3756	7388/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
segment_number	7452	7504/P	7505/P	7510/P	7522	7540/P	7541/P	7544/P	7551
segment_number	7573	7595	7596/P	7651					
segment_number	7680	7696/M	7707/P	7741/M	7890	7896/P	7898/P	7901	7901
		7921	7936/P	7946/P	7949/P	7952			
segment_number	7961	7970/P							
segment_number	8043	8054	8059/P	8067					
segment_number	8075	8086/P	8090						
segment_number	8309	8345	8346/P						
segment_number	8391	8403	8404	8412/P	8413/P				
segment_number	8425	8437/P	8444						
segment_number	8527	8542/P							
segment_number	8605	8615/P							
segment_number	8850	8885/P	8890	8917/P					
segment_number	9266	9289/M							
segment_number	9301	9358/M							
segment_number	9389	9393/M	9394/P	9411/P					
segment_number	9421	9437/P							
segment_number	9680	9691/P							
segment_number	9739	9755/P							
segment_number	9782	9798/P	9899						
segment_number	9986	9998	9999/P						
segment_number	10072	10083	10084	10092/P	10093/P				
segment_pointer	7665	7947/P	7950/P	7952/M					
segment_pointer	8650	8664/P	8694/P	8707					
segment_res_state	7291	7318							
segment_res_state	7681	7892/M	7894/M	7896/P					
segment_res_state	9310	9320/M	9321/P						
segment_reservation_state	4493	7318	7506/M	7506	8160	8162	8203	8254	8258/M
		9632	9634/M	9636/M					
segment_table_address_1	4788	8191/M							
segment_table_address_2	4790	8192/M							
segment_table_length	4784	7219	7221	7233	7305	7315	7901	7901	8054
		8158	8193/M	8298	8403	8754	8756	8939	9000
		9047	9531	9617	9627	9944	10083		
segment_table_length	7298	7305/M	7310	7315/M					
segment_table_length	9606	9617/M	9622	9627/M					
segment_table_size_increase	7198	7221							
segnum	3744	9015/M							
segnum	3841	7741	8444/M						
segnum	5915	5918							
segnum	5927	5930							
segnum	7292	7304/M	7309/M	7309	7310	7317/S	7318/S		
segnum	7329	7388							
segnum	7447	7504	7540						
segnum	7447	7505	7541						
segnum	7572	7596							
segnum	8042	8059							
segnum	8121	8249							
segnum	8121	8250	8259						
segnum	8143	8200	8201/S	8202/S	8203/S	8205/S	8208/S	8223	8227
		8228	8237/S	8239	8241/S	8249/P	8250/P	8254/S	8256/S
		8257/S	8259/P						
segnum	8289	8298	8299/S	8300/S	8301/P				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
segnum	8390	8412						
segnum	8390	8413						
segnum	8584	8586/M	8587/P					
segnum	8729	8765						
segnum	8849	8917						
segnum	8866	8939	8940/S	8941/S	8941	8942/S	8943/S	8944/S
segnum	8982	9001						
segnum	8982	9003						
segnum	8990	9000	9001/P	9003/P	9004	9005	9015	
segnum	9040	9047	9048/S	9048	9049/S	9050/S	9052/S	9054/S
segnum	9263	9284						
segnum	9263	9286						
segnum	9272	9278/P	9284/P	9286/P	9289			
segnum	9295	9329						
segnum	9295	9330						
segnum	9311	9321/P	9329/P	9330/P	9356/P	9358		
segnum	9516	9531	9533	9542/S	9543/S	9544/S	9547/P	
segnum	9600	9629						
segnum	9600	9630						
segnum	9611	9616/M	9621/M	9621	9622	9629/P	9630/P	9638
segnum	9931	9944	9945/S	9946/S	9949/S	9954/S	9955/S	
segnum	10071	10092						
segnum	10071	10093						
seq_pointer	3907	7947/P						
seqno	2076	5577	5621	5710/M	5735/M	5797	7514	7516/M	7612
		7645/M	7712	7717	7914/M	7955/M	8022/M	8094/M	8243
		8245/M	8417	8520/M	8552	8555/M	8599/M	8711/M	8896/M
		8907/M	8926	8929/M	8952/M	9338	9340/M	9535	9538/M
		9817	9819/M	10009/M	10019/M	10028/M			
sequential_accesses	4505	7925/M	9350/M	9467	9468/M	9715	9716/M		
sequential_random_selection	9299	9347							
server_file	3757	7389/M							
sfc$no_limit	1762	7111	7606	8595					
sfc$temp_file_space_limit	1763	8934/P							
sfid	3591	8356/M							
sfid	3742	7176/M	8799	9544/M	9545/P	9557/P			
sfid	3768	9250/M							
sfid	3806	7706	7716	7915					
sfid	4491	7510/P	7544/P	7599	7724/M	7870	8213/M	8232/M	8243/P
		8350	8417/P	8552/P	8566/P	8627/P	8697	8702/M	8959/P
		9006/P	9020/P	9334/M	9399/P	9411/P	9544	9770/P	9817/P
		9949/P	9954/P	10031/P	10037/M				
sfid	5533	5542	5543	5549	5550				
sfid	5789	5795/P							
sfid	5789	5795	5795	5795	5795				
sfid	6799	7360							
sfid	7169	7171/M	7176	7181/P					
sfid	7326	7360	7361	7362	7387				
sfid	7447	7514/P							
sfid	7447	7514	7514	7514	7514				
sfid	7572	7612/P							
sfid	7572	7612	7612	7612	7612				
sfid	7588	7599/M	7608/M	7612/P	7623/P	7638/P	7640/P		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
sfid	7662	7712/P	7717/P						
sfid	7662	7712	7712	7712	7712	7717	7717	7717	7717
sfid	7682	7707/P	7712/P	7716/M	7717/P	7724			
sfid	8121	8243/P							
sfid	8121	8243	8243	8243	8243				
sfid	8144	8221/P	8223/M	8232					
sfid	8327	8350/M	8352/M	8356					
sfid	8390	8417/P							
sfid	8390	8417	8417	8417	8417				
sfid	8526	8552/P							
sfid	8526	8552	8552	8552	8552				
sfid	8604	8627	8627	8627	8627				
sfid	8787	8799/M	8812/P						
sfid	8849	8926/P							
sfid	8849	8926	8926	8926	8926				
sfid	8982	9006	9006	9006	9006				
sfid	9295	9338/P							
sfid	9295	9338	9338	9338	9338				
sfid	9296	9334	9338/P	9356/P					
sfid	9507	9535/P							
sfid	9507	9535	9535	9535	9535	9545	9545	9545	9545
		9557	9557	9557	9557				
sfid	9738	9770	9770	9770	9770				
sfid	9781	9817/P							
sfid	9781	9817	9817	9817	9817				
sfid	9920	9949	9949	9949	9949				
sft$counter	6527	6201	6203	6204	6206	6519	6520		
sft$file_space_limit_kind	1762	491	2896	3804	4498	5365	5397	5475	5488
		7164	7582						
shadow_fde_p	7463	7514/P	7515/M	7515	7516/P				
shadow_fde_p	8867	8926/P	8927/M	8927	8928	8929/P	8931	8932	8933/P
shadow_info	4497	7513	7514/P	7855/M	7857/M	7864	7868/M	7869/M	7870/M
		7871/M	8209	8210	8211/M	8212/M	8213	8696/M	8697/M
		8698/M	8699/M	8925	8926/P	8933/P	8936	8941	8942
		8943/M	8944/M	10007	10037	10038/M	10039/M		
shadow_length	3867	7847	7856						
shadow_length_page_count	4582	7855/M	8699/M						
shadow_open_count	8868	8928/M	8930						
shadow_p	3866	7851	7858	7859/P	7869				
shadow_reference_info	4102	9130	9131/P	9132/P	9140/P	9141/P	9143/P	9143/P	9144/P
shadow_sdt_p	7684	7860/P							
shadow_sdtx_p	7685	7860/P	7864	7870	7871/M				
shadow_segment_kind	4584	7513	7864	7868/M	8209	8210	8211/M	8696/M	8925
		8942	8943/M	10007	10038/M				
shadow_segment_kind	8652	8689	8696						
shadow_segment_number	4586	7869/M	8941						
shadow_sfid	4583	7514/P	7870/M	8213	8697/M	8926/P	8933/P	10037	
shadow_start_page_number	4581	7857/M	8698/M						
shared_taskid_array	7451	7529	7537	7537	7538/P				
shared_taskid_array	7664	7697	7698	7698	7699/P	7891	7936/P		
shared_taskid_array	8852	8874	8875	8875	8876/P	8913	8914	8914	8915/P
size_offset	5946	5950/M	5952	5952	5953/S	5954/M	5954		
size_offset	8308	8338/M	8338	8338/S	8338/M	8338	8338		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
software_attri_set	3858	7783	7785	7788	7792	8497/M			
software_attribute_set	4494	7507	7521	7535/M	7535	7543	7600	7601	7783/M
		7792/M	7841/M	7841	7924	8229	8246/M	8246	8497
		8675	9348/M	9354	9454	9456/M	9456	9478/M	9478
		9703/M	9703	9706/M	9706	9709/M	9709	9714	9729
source	5818	5828							
source	9111	9144							
source_pva	4620	9131/P	9132/P	9140/P	9143/P				
st	4453	7317	8160	8161	8201	8205	8241/M	8256	8299
		8940	9048	9052/M	9054/M	9542	9945		
stack_for_ring	2017	7526/M	8230/M						
stack_pages_saved	4083	8037/M							
stack_pages_to_be_freed	8028	8037							
starting_byte_address	8529	8551	8566/P						
status	3559	7264/P							
status	3590	8362/P							
status	3717	8794	8795						
status	3775	9258/P							
status	6115	6118/M	6119	6120/P					
status	7165	7181/P							
status	7187	7264/M	7264	7264/P					
status	7189	7212/M	7234/P	7241/P	7242/P	7243	7264/P	7265	7266/P
		7276/P							
status	7293	7303/M	7311/P	7312					
status	7576	7592/M	7602/P	7623/P	7631	7638/P	7640/P	7650	
status	7666	7689/M	7701/P	7709/P	7719/P	7735/P	7752/P	7774/P	7789/P
		7795/P	7813/P	7821/P	7828/P	7837/P	7848/P	7852/P	7860/P
		7861	7865/P	7896/P	7898/P	7899	7899	7900/M	7902/P
		7903	7907	7908/P	7913				
status	7964	7970/P	7971	7976/P	7985/P				
status	7996	8005/P	8006	8011/P	8020/P				
status	8030	8035/M							
status	8044	8051/M	8055/P	8061/P					
status	8076	8084/M	8086/P						
status	8123	8148/M	8152/P	8176/P	8177/P	8178	8234/P		
status	8283	8292/M	8301/P						
status	8308	8362/M	8362	8362/P					
status	8314	8329/M	8346/P	8347	8362/P	8363			
status	8395	8400/M	8405/P	8407/P	8415/P				
status	8428	8436/M	8437/P	8516/P					
status	8533	8542/P	8543	8547/P	8567/P				
status	8577	8587/P	8588						
status	8608	8615/P	8616	8623/P					
status	8653	8664/P	8665	8676/P	8682/P	8694/P			
status	8786	8812/P	8813						
status	8853	8878/P	8885/P	8886	8897/P	8905/P	8906	8908/P	8934/P
		8959/P							
status	8991	8994/M	9020/P						
status	9035	9043/M							
status	9078	9089/P	9090	9091	9091	9092	9095/P	9096/P	9097
		9098	9100/P	9101					
status	9123	9141/P							
status	9156	9173/P	9181/P	9182					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
status	9238	9258/M	9258	9258/P					
status	9242	9247/M	9258/P						
status	9267	9275/M	9278/P	9279					
status	9302	9315/M	9321/P	9322					
status	9381	9394/P	9395	9399/P	9400	9411/P	9412	9412	9415
status	9427	9436/M	9437/P	9438	9445/P				
status	9509	9522/M	9561/M						
status	9603	9615/M	9623/P	9624					
status	9683	9691/P	9692	9699/P					
status	9742	9755/P	9756	9763/P	9766/P				
status	9785	9798/P	9799	9804/P	9821/P	9825/P	9838/P	9860/P	9869/P
		9880/P	9888/P	9894/P					
status	9922	9934/M	9937/P	9955/P	9965/P				
status	9987	9999/P	10000	10008/P	10017/P	10018	10020/P	10031/P	
status	10075	10080/M	10085/P	10087/P	10095/P				
ste	4440	7317	7351	7356/M	7478	7478	7479	7488/M	7493/M
		7507	7522	7525/S	7526	7543	7692/M	7693/M	7738/M
		7739/M	7744/M	7746/M	7750/M	7757/M	7759/M	7762/M	7764/M
		7767/M	7770/M	7777/M	7800/M	7806/M	7808/M	7816/M	7824/M
		7825/M	7835/M	7946/P	7949/P	7952	7975	8060	8069/M
		8092/M	8160	8161	8201	8206/M	8230	8241/M	8256
		8299	8414	8441	8442	8454	8457	8469	8478
		8490	8507	8512	8513	8514	8546	8546	8598/M
		8622	8674	8690/M	8706/M	8766	8918/M	8940	8963/M
		9002	9021/M	9048	9052/M	9054/M	9336/M	9337/M	9342/M
		9345/M	9354	9444	9444	9542	9631	9698	9727
		9762	9765	9814/M	9815/M	9830/M	9832/M	9836/M	9843/M
		9845/M	9848/M	9850/M	9853/M	9856/M	9863/M	9873/M	9875/M
		9883/M	9891/M	9892/M	9945	10036/M	10094		
ste	8738	8766/M	8767	8767					
ste_p	7327	7352/M	7357/M	7374/M					
stl	8146	8158/M	8160/S	8160/S	8160/S	8160/S	8160	8161/S	8162/S
		8163	8164/M	8164	8167	8169	8170	8175	8188
		8193	8200						
stlc_allocation	4063	5710	5710/M	5740	5741/M	7516	7516/M	7645	7645/M
		7914	7914/M	7955	7955/M	8022	8022/M	8094	8094/M
		8245	8245/M	8520	8520/M	8555	8555/M	8599	8599/M
		8711	8711/M	8896	8896/M	8907	8907/M	8929	8929/M
		8952	8952/M	9085/M	9340	9340/M	9538	9538/M	9819
		9819/M	10009	10009/M	10019	10019/M	10028	10028/M	
store_ste_in_segment_table	7324	7394	7510	7544	9356				
str1	5823	5828/M	5830						
str1	9111	9144/M	9144						
str2	5824	5829/M	5830/M	5831					
str2	9111	9144/M	9144/M	9144					
stream	4499	7625	7627/M	7878/M	7879/M	7925/M	7928	7933/M	9350/M
		9353/M	9459	9462	9463/M	9465	9466/M	9467	9468/M
		9476/M	9481	9482/M	9715	9716/M			
streaming	4508	9465							
stt$set_name	1624	503	1554	1559					
subfunction_code	3794	5520/M	8446/M	8628/M	9772/M				
syc$rc_change_segment_table	3634	7259							
syc$rc_cycle	3621	5577	5619	5710	5746	5769	5797	7514	7516

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		7612	7645	7712	7717	7914	7955	8022	8094
		8243	8245	8417	8520	8552	8555	8599	8711
		8896	8907	8926	8929	8952	9338	9340	9535
		9538	9817	9819	10009	10019	10028		
syc$rc_fetch_offset_mod_pages	3688	8355							
syc$rc_ring1_segment_request	3653	7174	7385	9013	9525				
syc$rc_ring1_server_seg_request	3697	9249							
syc$rc_set_get_segment_length	3650	5519	8446	8628	9771				
syc$ucr_condition	5077	5088							
syc$user_defined_condition	5078	5090							
syc$volume_unavailable	5081	9169/P							
syp$cause_condition	6108	9169							
syp$cycle_for_lock	5754	5710	5746	5776	7516	7645	7914	7955	8022
		8094	8245	8520	8555	8599	8711	8896	8907
		8929	8952	9340	9538	9819	10009	10019	10028
syp$mfh_for_hang_task	6112	9567							
syp$return_jobs_ri_resources	6126	9028							
syp$set_status_from_mtr_status	6113	6123	7264	8362	9258				
syp$terminate_task	6128	9563							
system_file_id	3755	7387/M							
system_file_id	9239	9250							
system_give_up_cpu	4073	5710	5745	7516	7645	7914	7955	8022	8094
		8245	8520	8555	8599	8711	8896	8907	8929
		8952	9340	9538	9819	10009	10019	10028	
system_table_lock_count	4068	5577/M	5577	5577/M	5577	5611/M	5611	5618/M	5618
		5710/M	5710	5710	5710	5738/M	5738	5740	5745
		5797/M	5797	5797/M	5797	7514/M	7514	7514/M	7514
		7516/M	7516	7516	7516	7612/M	7612	7612/M	7612
		7645/M	7645	7645	7645	7712/M	7712	7712/M	7712
		7717/M	7717	7717/M	7717	7914/M	7914	7914	7914
		7955/M	7955	7955	7955	8022/M	8022	8022	8022
		8094/M	8094	8094	8094	8243/M	8243	8243/M	8243
		8245/M	8245	8245	8245	8417/M	8417	8417/M	8417
		8520/M	8520	8520	8520	8552/M	8552	8552/M	8552
		8555/M	8555	8555	8555	8599/M	8599	8599	8599
		8711/M	8711	8711	8711	8896/M	8896	8896	8896
		8907/M	8907	8907	8907	8926/M	8926	8926/M	8926
		8929/M	8929	8929	8929	8952/M	8952	8952	8952
		9084	9338/M	9338	9338/M	9338	9340/M	9340	9340
		9340	9535/M	9535	9535/M	9535	9538/M	9538	9538
		9538	9817/M	9817	9817/M	9817	9819/M	9819	9819
		9819	10009/M	10009	10009	10009	10019/M	10019	10019
		10019	10028/M	10028	10028	10028			
syt$monitor_flag	4984	4969							
syt$monitor_flags	4969	4056							
syt$monitor_request_code	3604	3558	3589	3716	3767	3791	5691		
syt$monitor_status	3569	3559	3590	3717	3775	6114			
task_id	5572	5577/M	5577	5577					
task_id	5599	5604/M	5606	5614					
task_id	5706	5710/M	5710						
task_id	5721	5726/M	5728						
task_id	5789	5797/M	5797	5797					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
task_id	7447	7514/M	7514	7514					
task_id	7447	7516/M	7516						
task_id	7572	7612/M	7612	7612					
task_id	7572	7645/M	7645						
task_id	7662	7712/M	7712	7712	7717/M	7717	7717		
task_id	7662	7914/M	7914	7955/M	7955				
task_id	7992	8022/M	8022						
task_id	8074	8094/M	8094						
task_id	8121	8243/M	8243	8243					
task_id	8121	8245/M	8245						
task_id	8122	8150/P							
task_id	8390	8417/M	8417	8417					
task_id	8424	8520/M	8520						
task_id	8526	8552/M	8552	8552					
task_id	8526	8555/M	8555						
task_id	8573	8599/M	8599						
task_id	8649	8711/M	8711						
task_id	8849	8896/M	8896	8907/M	8907	8929/M	8929	8952/M	8952
task_id	8849	8926/M	8926	8926					
task_id	9295	9338/M	9338	9338					
task_id	9295	9340/M	9340						
task_id	9507	9535/M	9535	9535					
task_id	9507	9538/M	9538						
task_id	9781	9817/M	9817	9817					
task_id	9781	9819/M	9819						
task_id	9921	9935/P							
task_id	9985	10009/M	10009	10019/M	10019	10028/M	10028		
task_sdt_entry_p	8871	8917/M	8918/M						
task_sdt_p	7464	7540/M	7544/P						
task_sdtx_p	7465	7541/M	7542/M	7543/P					
task_xcb	7466	7538/P	7539	7540/P	7541/P				
task_xcb	7686	7698/P	7700						
task_xcb	8870	8876/P	8877	8915/P	8916	8917/P			
temp_pages_to_touch	5943	5948/M	5953	5957					
temp_pages_to_touch	8308	8338/M	8338	8338					
temp_transfer_size	9424	9460/P							
tmc$broken_task_fault_id	5005	5055							
tmc$btc_invalid_a0	5103	5124							
tmc$btc_invalid_p	5103	5124							
tmc$btc_mcr_traps_disabled	5104	5125							
tmc$btc_mf_traps_disabled	5103	5123							
tmc$btc_mntr_fault_buffer_full	5102	5123							
tmc$btc_system_error	5105	5119							
tmc$btc_ucr_traps_disabled	5104	5125							
tmc$cyc_clear_sys_lock	5699	5710/P	5746/P	7516/P	7645/P	7914/P	7955/P	8022/P	8094/P
		8245/P	8520/P	8555/P	8599/P	8711/P	8896/P	8907/P	8929/P
		8952/P	9340/P	9538/P	9819/P	10009/P	10019/P	10028/P	
tmc$cyc_set_fde_lock	5701	5577	5620	5797	7514	7612	7712	7717	8243
		8417	8552	8926	9338	9535	9817		
tmc$cycle_reason	5698	5692	5754						
tmc$dummy_fault	5006	5061							
tmc$flag_available_31	5240	5244							
tmc$maximum_monitor_faults	5010	5001							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$maximum_signals	5220	5217							
tmc$maximum_system_task_id	5253	5256							
tmc$mcr_fault	5005	5057							
tmc$signal_available_63	5202	5213							
tmc$stid_null_task	5259	5256							
tmt$broken_task_condition	5102	5118							
tmt$broken_task_monitor_fault	5116	5056							
tmt$mcr_faults	5141	5058							
tmt$monitor_fault_buffer	4995	4093							
tmt$monitor_fault_buffers	5001	4996	4997	4998					
tmt$monitor_fault_identifiers	5004	5054	5130						
tmt$rb_cycle	5690	5598	5767						
tmt$signal	5158	5153							
tmt$signal_buffer	5150	4094							
tmt$signal_buffers	5217	5151	5152	5153					
tmt$system_flags	5223	4069							
tmt$system_task_id	5256	4060							
tmt$task_queue_link	2102	2095	6832						
tos_registers	4796	7525/M	8997						
transfer_size	4506	7627/M	7878/M	7933/M	9353/M	9459	9462	9463/M	9481
		9482/M							
transfer_size_specified	4509	7625	7879/M	7928					
transfer_unit_size	2010	7626/P	7929/P						
ts	5869	5876/M	5878	5878	5880/M	5880			
ts	7572	7626/M	7626	7626/M	7626	7626			
ts	7662	7877/M	7877	7877/M	7877	7877	7929/M	7929	7929/M
		7929	7929	7931/M	7931	7931/M	7931	7931	
ts	9295	9352/M	9352	9352/M	9352	9352			
ts	9420	9460/M	9460	9460/M	9460	9460			
tstatus	9519	9537/P	9547/P	9548	9552	9553	9554	9554	9555
		9556	9561	9562					
user_attributes	3807	7728	7729	7729	7730	7733	7733	7734	7738
		7739	7741	7743	7748	7756	7761	7766	7769
		7772	7783	7785	7787	7792	7798	7800	7803
		7805	7811	7816	7819	7824	7825	7832	7833
		7834	7835	7843	7845	7847	7851	7856	7857
		7859/P	7869	7873					
user_attributes	7726	7726	7736	7753	7775	7790	7796	7814	7822
		7829	7838	7849	7853	7862	7866	7881	7904
		7911							
validating_ring	7962	7975							
validating_ring	8528	8546							
validating_ring	9783	9803	9824	9835	9859	9879	9887		
validating_ring_number	3803	7692	7693	7694	7749	7773	7782	7784	7786
		7812	7820	7827					
validating_ring_number	7995	8010							
validating_ring_number	8606	8622							
validating_ring_number	8651	8674							
validating_ring_number	8851	8894							
validating_ring_number	9422	9444	9444						
validating_ring_number	9681	9698							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
validating_ring_number	9740	9762							
vl	4467	7317	7488/M	7493/M	7762/M	7764/M	7806/M	7808/M	8060
		8069/M	8092/M	8160	8161	8201	8241/M	8256	8299
		8414	8490	8507	8767	8918/M	8940	8963/M	9002
		9021/M	9048	9052/M	9054/M	9342/M	9542	9631	9848/M
		9850/M	9873/M	9875/M	9945	10094			
wired_segment_length	3869	7843							
wp	4470	7479	7507	7543	7757/M	7759/M	7825/M	8469	8514
		8690/M	9345/M	9354	9671	9727	9765	9843/M	9845/M
		9892/M							
write_privilege	3926	7825	8514/M	9892					
xasid	7129	7138	7138						
xasti	7130	7144/M							
xasti	7148	7154							
xasti	7324	7355							
xcb	6010	6023/M							
xcb	7447	7472/M							
xcb	7572	7594/M							
xcb	7662	7888/M							
xcb	8027	8036/M							
xcb	8042	8052/M							
xcb	8121	8156/M							
xcb	8282	8293/M							
xcb	8390	8401/M							
xcb	8729	8749/M							
xcb	8849	8912/M	8937/M						
xcb	8982	8996/M							
xcb	9033	9044/M							
xcb	9072	9083/M							
xcb	9111	9126/M							
xcb	9263	9276/M							
xcb	9295	9316/M							
xcb	9507	9527/M							
xcb	9600	9613/M							
xcb	10071	10081/M							
xcb_p	5572	5577/M	5577	5577/M	5577	5577/M	5577		
xcb_p	5600	5602/M	5604	5611/M	5611	5618/M	5618		
xcb_p	5706	5710/M	5710	5710/M	5710	5710	5710	5710/M	5710
		5710							
xcb_p	5722	5724/M	5726	5738/M	5738	5740	5740	5741/M	5745
		5745							
xcb_p	5789	5797/M	5797	5797/M	5797	5797/M	5797		
xcb_p	5900	5906	5906	5907	5907				
xcb_p	5914	5917	5918						
xcb_p	5926	5929	5930						
xcb_p	6047	6048/M	6050						
xcb_p	7188	7217	7218	7219	7221	7233			
xcb_p	7289	7306	7306	7306	7306				
xcb_p	7290	7305	7306/P	7311/P	7315				
xcb_p	7447	7504	7504	7540	7540				
xcb_p	7447	7505	7505	7541	7541				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	7447	7514/M	7514	7514/M	7514	7514/M	7514		
xcb_p	7447	7516/M	7516	7516/M	7516	7516	7516	7516/M	7516
		7516							
xcb_p	7467	7472/P	7504/P	7505/P	7525/M	7539			
xcb_p	7572	7596	7596						
xcb_p	7572	7612/M	7612	7612/M	7612	7612/M	7612		
xcb_p	7572	7645/M	7645	7645/M	7645	7645	7645	7645/M	7645
		7645							
xcb_p	7589	7594/P	7596/P	7654/M					
xcb_p	7662	7712/M	7712	7712/M	7712	7712/M	7712	7717/M	7717
		7717/M	7717	7717/M	7717				
xcb_p	7662	7914/M	7914	7914/M	7914	7914	7914	7914/M	7914
		7914	7955/M	7955	7955/M	7955	7955	7955	7955/M
		7955	7955						
xcb_p	7687	7888/P	7896/P	7901	7901	7902/P	7922		
xcb_p	7992	8022/M	8022	8022/M	8022	8022	8022	8022/M	8022
		8022							
xcb_p	8033	8036/P	8037/M						
xcb_p	8042	8059	8059						
xcb_p	8049	8052/P	8054	8059/P					
xcb_p	8074	8094/M	8094	8094/M	8094	8094	8094	8094/M	8094
		8094							
xcb_p	8121	8159	8159	8159	8159				
xcb_p	8121	8243/M	8243	8243/M	8243	8243/M	8243		
xcb_p	8121	8245/M	8245	8245/M	8245	8245	8245	8245/M	8245
		8245							
xcb_p	8121	8249	8249						
xcb_p	8121	8250	8250	8259	8259				
xcb_p	8282	8297	8297	8297	8297				
xcb_p	8290	8293/P	8297/P	8298					
xcb_p	8390	8412	8412						
xcb_p	8390	8413	8413						
xcb_p	8390	8417/M	8417	8417/M	8417	8417/M	8417		
xcb_p	8398	8401/P	8403	8412/P	8413/P				
xcb_p	8424	8520/M	8520	8520/M	8520	8520	8520	8520/M	8520
		8520							
xcb_p	8526	8552/M	8552	8552/M	8552	8552/M	8552		
xcb_p	8526	8555/M	8555	8555/M	8555	8555	8555	8555/M	8555
		8555							
xcb_p	8573	8599/M	8599	8599/M	8599	8599	8599	8599/M	8599
		8599							
xcb_p	8649	8711/M	8711	8711/M	8711	8711	8711	8711/M	8711
		8711							
xcb_p	8729	8765	8765						
xcb_p	8739	8749/P	8754	8756	8765/P				
xcb_p	8849	8896/M	8896	8896/M	8896	8896	8896	8896/M	8896
		8896	8907/M	8907	8907/M	8907	8907	8907	8907/M
		8907	8907	8929/M	8929	8929/M	8929	8929	8929
		8929/M	8929	8929	8952/M	8952	8952/M	8952	8952
		8952	8952/M	8952	8952				
xcb_p	8849	8917	8917						
xcb_p	8849	8926/M	8926	8926/M	8926	8926/M	8926		
xcb_p	8849	8938	8938	8938	8938				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	8872	8912/P	8916	8937/P	8938/P	8939			
xcb_p	8982	9001	9001						
xcb_p	8982	9003	9003						
xcb_p	8992	8996/P	8997	9000	9001/P	9003/P			
xcb_p	9033	9046	9046	9046	9046				
xcb_p	9041	9044/P	9046/P	9047					
xcb_p	9072	9099/M	9099						
xcb_p	9079	9083/P	9084	9085/M					
xcb_p	9124	9126/P	9130	9131/P	9132/P	9140/P	9141/P	9143/P	9143/P
		9144/P							
xcb_p	9149	9180/M	9180						
xcb_p	9263	9284	9284						
xcb_p	9263	9286	9286						
xcb_p	9273	9276/P	9278/P	9284/P	9286/P				
xcb_p	9295	9329	9329						
xcb_p	9295	9330	9330						
xcb_p	9295	9338/M	9338	9338/M	9338	9338/M	9338		
xcb_p	9295	9340/M	9340	9340/M	9340	9340	9340	9340/M	9340
		9340							
xcb_p	9313	9316/P	9321/P	9329/P	9330/P				
xcb_p	9507	9528	9528	9528	9528				
xcb_p	9507	9535/M	9535	9535/M	9535	9535/M	9535		
xcb_p	9507	9538/M	9538	9538/M	9538	9538	9538	9538/M	9538
		9538							
xcb_p	9520	9527/P	9528/P	9531	9534	9535/P	9537/P		
xcb_p	9600	9629	9629						
xcb_p	9600	9630	9630						
xcb_p	9610	9613/P	9617	9623/P	9627	9629/P	9630/P		
xcb_p	9781	9817/M	9817	9817/M	9817	9817/M	9817		
xcb_p	9781	9819/M	9819	9819/M	9819	9819	9819	9819/M	9819
		9819							
xcb_p	9920	9941	9941	9941	9941				
xcb_p	9932	9935/P	9936	9941/P	9944	9961	9962		
xcb_p	9985	10009/M	10009	10009/M	10009	10009	10009	10009/M	10009
		10009	10019/M	10019	10019/M	10019	10019	10019	10019/M
		10019	10019	10028/M	10028	10028/M	10028	10028	10028
		10028/M	10028	10028					
xcb_p	10071	10092	10092						
xcb_p	10071	10093	10093						
xcb_p	10078	10081/P	10083	10092/P	10093/P				
xp	4055	7219	7221	7233	7305	7315	7525/M	7901	7901
		8054	8158	8191/M	8192/M	8193/M	8298	8403	8754
		8756	8939	8997	9000	9047	9531	9617	9627
		9944	10083						
xp	4468	7478	7478	7767/M	7770/M	7777/M	7816/M	8478	8512
		8767	9663	9853/M	9856/M	9863/M	9883/M		
xsdt_entry	7325	7349	7374						
xsfid	7162	7171							
xsfid	7574	7608							
xsfid	8310	8352							

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

System Constant Management

```
    3 MODULE sym$system_constant_manager;
    4
    5 { PURPOSE:
    6 {   This module contains procedures that allow the operator to fetch and
    7 {   modify the values of system constants during deadstart or during system
    8 {   operation.  The priviledge of modifying the constants is restricted to the
    9 {   system job.
   10
```


System Constant Management
  Global Declarations Referenced by this module

```
o  1618
o  1619       PROCEDURE [XREF] dmp$fetch_debug_option_value (name: string(*);
o  1620             VAR value: integer;
o  1621             VAR status: ost$status);
o  1622
   1673
   1674     PROCEDURE [XREF] dmp$store_debug_option_value (name: string (*);
   1675                 value: integer;
   1676           VAR status: ost$status);
   1677
o  1680
o  1681   PROCEDURE [XREF] iop$fetch_debug_option_value (name: string ( * );
o  1682     VAR value: integer;
o  1683     VAR status: ost$status);
o  1684
   1687
   1688   PROCEDURE [XREF] iop$store_debug_option_value (name: string ( * );
   1689         value: integer;
   1690     VAR status: ost$status);
   1691
o  1694
o  1695    PROCEDURE [XREF] osp$append_status_parameter ALIAS 'ospasp'
o  1696      (     delimiter: char;
o  1697            text: string ( * <= osc$max_string_size);
o  1698       VAR status (input, output) : ost$status);
o  1699
   1702
   1703    PROCEDURE [XREF] osp$set_status_abnormal ALIAS 'ospssa'
   1704      (     identifier: ost$status_identifier;
   1705            condition: ost$status_condition_code;
   1706            text: string ( * <= osc$max_string_size);
   1707       VAR status: ost$status);
   1708
o  1715
o  1716    PROCEDURE [XREF] osp$system_error (error_message: string ( * );
o  1717          status: ^ost$status);
o  1718
   1721    VAR
   1722      bav$force_direct_tape_io: [XREF] boolean;
   1723    VAR
   1724      bav$max_allowed_tape_block_size: [XREF] integer;
   1725    VAR
   1726      bav$max_bytes_per_tape_io: [XREF] integer;
   1727    VAR
   1728      bav$max_indirect_tape_block: [XREF] integer;
   1729    VAR
   1730      bav$use_assign_pages_for_tape: [XREF] boolean;
   1731
   1732    VAR
   1733      dfv$file_server_debug_enabled: [XREF] boolean;
   1734
   1735    VAR
   1736      dfv$file_server_info_enabled: [XREF] boolean;
   1737
   1738    VAR
```

System Constant Management
Global Declarations Referenced by this module

```
1739      dfv$job_recovery_enabled: [XREF] boolean;
1740
1741    VAR
1742      { This is an integer to allow use with sym$system_constant_manager.
1743      { The true value range is 1 .. dfc$max_job_list_p_array_size
1744      dfv$maximum_client_job_lists: [XREF] integer;
o 3204
o 3205    VAR
o 3206      { This is an integer to allow use with sym$system_constant_manager
o 3207      { The true value range is dft$family_pointer_index
o 3208      dfv$number_served_family_lists: [XREF] integer;
3211
3212    VAR
3213      dmv$permanent_file_overflow: [XREF] boolean;
3214
3215    VAR
3216      dmv$recycle_device_log: [XREF] boolean;
3217
3218    VAR
3219      dmv$temporary_file_overflow: [XREF] boolean;
3220
3221    VAR
3222      dmv$test_recovery: [XREF] boolean;
3223
3224    VAR
3225      dpv$enable_console_bell: [XREF] boolean;
3226
3227
3228    VAR
3229      dsv$unload_deadstart_tape: [XREF] boolean;
3230
3231 {Job Control Block (JCB).}
3232
3233    VAR
3234      jmv$jcb: [XREF] jmt$job_control_block;
o 4138
o 4139    VAR
o 4140      jmv$maximum_job_classes: [XREF] 0..0ffffffff(16);
o 4141
o 4142
o 4143    VAR
o 4144      jmv$maximum_known_jobs: [XREF, oss$mainframe_pageable] ost$halfword;
o 4145
4151
4152    VAR
4153      jmv$maximum_known_outputs: [XREF, oss$mainframe_pageable] ost$halfword;
4154
o 4157
o 4158    VAR
o 4159      jmv$maximum_service_classes: [XREF] 0..0ffffffff(16);
o 4160
o 4161
o 4162    VAR
o 4163      jmv$scan_idle_dispatch_interval: [XREF] integer;
o 4164
```


System Constant Management
Global Declarations Referenced by this module

```
o 4165    VAR
o 4166      jmv$sched_memory_wait_factor: [XREF] integer;
o 4167
o 4168
o 4169    VAR
o 4170      jmv$scheduler_wait_time: [XREF] integer;
o 4171 {Define boolean that specifies whether jobs that go into long wait should be
o 4172 {swapped immediately.
o 4173
o 4174    VAR
o 4175      jmv$swap_jobs_in_long_wait: [XREF] boolean;
o 4176
o 4177 {Define value of AJL ORDINAL used by the system job
o 4178
o 4179    VAR
o 4180      jmv$system_ajl_ordinal: [XREF]jmt$ajl_ordinal;
o 4181
4184    VAR
4185      jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
4186
o 4189
o 4190    VAR
o 4191      jsv$enable_debug_code: [XREF] boolean;
o 4192
o 4193
o 4194     VAR
o 4195       jsv$enable_swap_file_statistics: [XREF] boolean;
o 4196
o 4197
o 4198    VAR
o 4199      jsv$halt_on_swapin_failure: [XREF] boolean;
o 4200
o 4201    VAR
o 4202      jsv$free_working_set_on_swapout: [XREF] boolean;
o 4203
o 4204
o 4205     VAR
o 4206       jsv$maximum_pages_to_swap: [XREF] integer;
o 4207
o 4208    VAR
o 4209      jsv$max_pages_first_swap_task: [XREF] integer;
o 4210
o 4211    VAR
o 4212      jsv$max_time_swap_io_complete: [XREF] integer;
o 4213
o 4214    VAR
o 4215      jsv$max_time_swap_io_not_init: [XREF] integer;
o 4216
o 4217
o 4218    VAR
o 4219      jsv$think_expiration_time: [XREF] integer;
o 4220
o 4221 {Define minimum number of pages that must be kept in the free + available page
o 4222 {queues. If the actual number drops below this value, memory manager begins
o 4223 {an aggressive aging policy. If the number of page frames drops below mmv$aggressive_aging_level_2
```

System Constant Management
Global Declarations Referenced by this module

```
o  4224 {then only critical system tasks are assigned memory. User tasks are put into a memory wait queue.
o  4225
o  4226   VAR
o  4227     mmv$aggressive_aging_level: [XREF] integer,
o  4228     mmv$aggressive_aging_level_2: [XREF] integer;
o  4229
o  4230 {The following variable defines the agining algorithm that is used by memory manager.
o  4231 {     0 - no swapping active
o  4232 {     1 - swapping active
o  4233 {   > 1 - to be defined
o  4234   VAR
o  4235     mmv$aging_algorithm: [XREF] integer;
o  4236
o  4237   VAR
o  4238     mmv$aio_limit: [XREF] integer;
o  4239
o  4240 { Define variable used to activate/deactivate assign_multiple_pages. When activated
o  4241 { mmp$page_pull will assign more than one page to a task that has faulted for a new page
o  4242 { and it is determined that extra pages will highly likely be used.  Multiple pages will
o  4243 { be assigned if mmv$reassignable_page_frames.now is greater or equal to this value.
o  4244
o  4245   VAR
o  4246     mmv$assign_multiple_pages: [XREF] integer;
o  4247 { -------- Declarations for forcing the use of cache and maps omitted at compile time --------
o  4248
o  4249
o  4250   VAR
o  4251     mmv$disable_write_for_perf_meas: [XREF] boolean;
o  4252
o  4253  VAR
o  4254    mmv$file_allocation_interval: [XREF] integer;
o  4255 { -------- Declarations for forcing the use of cache and maps omitted at compile time --------
o  4256 {Define number of page frames that memory manager should attempt to
o  4257 {keep in the available + free queues.
o  4258
o  4259   VAR
o  4260     mmv$free_queue_threshold: [XREF] integer;
o  4261
o  4262 { Global Page Queue List array.
o  4263
o  4264   VAR
o  4265     mmv$gpql: [XREF] mmt$global_page_queue_list;
   4268
   4269 {This variable defines the rate at which memory manager scans all jobs and
   4270 {ages the working sets of any CP bound job that is found.
   4271
   4272   VAR
   4273     mmv$jws_queue_age_interval: [XREF] integer;
   4274
   4275
   4276 { The data consists of pointers to the various mmv$ variables managed by MMU and also the default value
   4277 { for each of the variables and the default value of the Global Page Queue List.  The default values
   4278 { are saved by mmm$deadstart_initialization.
   4279
   4280   VAR
```


System Constant Management
Global Declarations Referenced by this module

```
   4281     mmv$manage_memory_utility: [XREF] mmt$manage_memory_utility;
o  4316
o  4317 {Maximum number of pages that will be assigned to a segment that does not
o  4318 {have a backing file. A signal is sent to the task to assign a file when
o  4319 {the number of assigned pages exceeds this value.
o  4320
o  4321   VAR
o  4322     mmv$max_pages_no_file: [XREF] integer;
o  4323
o  4324   VAR
o  4325     mmv$min_avail_pages: [XREF] integer;
o  4326
o  4327 {Define variable used to activate/deactivate multi-page write. Multi-page write activated
o  4328 {means that memory manager attempts to write all modified pages in a transfer unit whenever
o  4329 {any page in the transfer unit is written.
o  4330
o  4331   VAR
o  4332     mmv$multi_page_write: [XREF] boolean;
o  4333
o  4334 {Define option used to disable buffering of aged out pages in memory, i.e. when a page
o  4335 {is aged out, it is written to disk (if necessary) and placed in the free queue.
o  4336
o  4337   VAR
o  4338     mmv$no_memory_buffering: [XREF] boolean;
o  4339
o  4340 {This variable the time interval between calls from CP Monitor to
o  4341 {memory manager MMP$PERIODIC_CALL procedure.
o  4342
o  4343   VAR
o  4344     mmv$periodic_call_interval: [XREF] integer;
o  4345   VAR
o  4346     mmv$read_tu_read_write: [XREF] 0 .. 0ffffffff(16),
o  4347     mmv$read_tu_execute: [XREF] 0 .. 0ffffffff(16);
o  4348
o  4349
o  4350     VAR
o  4351       mmv$shared_pages_in_jws: [XREF] boolean;
o  4352
o  4353 {This variable the rate at which Memory Manager ages out all pages of all
o  4354 {working sets that have not been referenced since the last time the
o  4355 {aging was done.
o  4356
o  4357   VAR
o  4358     mmv$shared_queue_age_interval: [XREF] integer;
o  4359 {Define the fundemental page aging constant.
o  4360
o  4361   VAR
o  4362     mmv$tick_time: [XREF] integer;
o  4363
o  4364
o  4365 {Define option used to force aged out pages to be written to disk
o  4366 {immediately when it is aged out of a job working set.
o  4367 {Page are written if SOON + NOW < mmv$write_aged_out_pages.
o  4368
o  4369   VAR
```

System Constant Management
Global Declarations Referenced by this module

```
o  4370      mmv$write_aged_out_pages: [XREF] integer;
o  4371
o  4372
o  4373    VAR
o  4374      mtv$halt_on_proc_malf: [XREF] boolean;
o  4375
o  4376    VAR
o  4377      mtv$aborted_task_threshold: [XREF] integer;
o  4378
o  4379
o  4380    VAR
o  4381      mtv$automatic_unstep_resume: [XREF] boolean;
o  4382
o  4383  {If a MCR fault occurs in a task executing in a ring <= the value
o  4384  {of this variable, Monitor will halt the CPU. The variable is located
o  4385  {in Monitor and is intended to be used for debug only.
o  4386
o  4387    VAR
o  4388      mtv$system_haltring: [XREF] 0 .. 255,
o  4389      mtv$halt_cpu_ring_number: [XREF] 0 .. 255;
o  4390
o  4391    VAR
o  4392      mtv$processor_due_threshold: [XREF] integer;
o  4393
o  4394  VAR
o  4395      mtv$sys_core_init_complete: [XREF] boolean;
o  4396  {At deadstart time, monitor copies the job Exchange Package to this location
o  4397  {before the first exchange to job mode.
o  4398
o  4399    VAR
o  4400      mtv$xp_initial_value: [XREF] ost$exchange_package;
o  4401
   4594  {System page size.}
   4595
   4596    VAR
   4597      osv$page_size: [XREF] ost$page_size;
   4598
o  4601
o  4602    VAR
o  4603      osv$global_processor_model_info: [XREF] ost$processor_model_definition;
o  4604
   4693
   4694    VAR
   4695      osv$special_aam_trap: [XREF] boolean;
   4696
   4697    VAR
   4698      pmv$quantum: [XREF] integer;
   4699
   4700     VAR
   4701       syv$enable_heap_trace: [XREF] boolean;
   4702
   4703  { Variable which determines number of lines displayed on the console during
   4704  { a SYSDEBUG session display command.
   4705
   4706    VAR
```


System Constant Management
Global Declarations Referenced by this module

```
   4707      syv$debugger_page_wait_lines: [XREF] integer;
   4708
   4709
   4710    VAR
   4711      syv$dflt_debug_output_disposal: [XREF] syt$dflt_debug_output_disposal;
   4712
o  4733
o  4734  { The following boolean will be set TRUE by SETSA to enable the fault injection utility.
o  4735
o  4736    VAR
o  4737      syv$enable_fault_injection: [XREF] boolean;
o  4738    VAR
o  4739      syv$job_recovery_option: [XREF] integer;
o  4740  CONST
o  4741    syc$jre_enabled = 0,
o  4742
o  4743    syc$jre_command_disabled = 3,
o  4744
o  4745    syc$jre_prior_ds_disabled = 5,
o  4746    syc$jre_no_image = 6,
o  4747    syc$jre_different_system = 7,
o  4748    syc$jre_page_size_mismatch = 8,
o  4749
o  4750    syc$jre_system_disabled =9;
o  4751
o  4752
o  4753    VAR
o  4754      syv$mandatory_dualstate: [XREF] boolean;
o  4755
o  4756
o  4757    VAR
o  4758      syv$max_debug_output_lines: [XREF] integer;
o  4759
o  4760
o  4761     VAR
o  4762       syv$verify_heap_linkage: [XREF] boolean;
o  4763  {Define length of timeout that occurs when a user issues a 'CYCLE' request.
o  4764
o  4765    VAR
o  4766      tmv$cycle_delay_time: [XREF] integer;
o  4767
o  4768
o  4769    VAR
o  4770      tmv$dedicate_a_cpu_to_nos: [XREF] boolean;
o  4771  {Define variables that control action taken by monitor when system errors occur.
o  4772
o  4773    VAR
o  4774      tmv$halt_on_hung_task: [XREF] boolean,
o  4775      tmv$system_debug_ring: [XREF] integer,
o  4776    tmv$system_debug_segment: [XREF] integer,
o  4777      tmv$system_error_hang_count: [XREF] 0 .. 0ffffffff(16);
o  4778
o  4779
o  4780    VAR
o  4781      tmv$max_idle_sit_value: [XREF] integer;
```

System Constant Management
Global Declarations Referenced by this module

```
o  4782
o  4783   VAR
o  4784     tmv$timed_wait_not_queued: [XREF] integer;
o  4785
o  4786
```


System Constant Management
Global Declarations Declared by this module

```
o  4788
o  4789   TYPE
o  4790     value_table_type = record
o  4791       deadstart_only: boolean,
o  4792       name: string (max_name_length),
o  4793       min,
o  4794       max: integer,
o  4795       case value_type: (v_integer, v_halfword, v_byte, v_boolean) of
o  4796       = v_integer =
o  4797         integer_p: ^integer,
o  4798       = v_halfword =
o  4799         halfword_p: ^0 .. 0ffffffff(16),
o  4800       = v_byte =
o  4801         byte_p: ^0 .. 255,
o  4802       = v_boolean =
o  4803         boolean_p: ^boolean,
o  4804       casend,
o  4805     recend;
o  4806
o  4807   VAR
o  4808     clv$command_statistics_enabled: [XDCL, #GATE] boolean := FALSE,
o  4809     dmv$space_messages_to_console: [XDCL, #GATE] boolean := TRUE,
o  4810     dmv$trim_files: [XDCL, #GATE] boolean := TRUE,
o  4811     dmv$volume_class_kludge: [XDCL] boolean := FALSE,
o  4812     dpv$display_delay: [XDCL, #GATE] integer := 1000,
o  4813     dsv$display_deadstart_messages: [XDCL] boolean := TRUE,
o  4814     dsv$load_files: [XDCL, #GATE] boolean := FALSE,
o  4815     dsv$ignore_image: [XDCL] boolean := FALSE,
o  4816     iiv$condition_handler_trace: [XDCL, #GATE] boolean := false,
o  4817     iiv$interactive_wait_time: [XDCL, #GATE] integer := 60000000,
o  4818     iiv$output_option: [XDCL, #GATE] integer := 0,
o  4819     iiv$suspended_job_timeout: [XDCL, #GATE] integer := 180 * 60 * 1000,
o  4820     iov$unused_boolean: [XDCL, #GATE] boolean := TRUE,
o  4821     jmv$enable_queue_file_access: [XDCL, #GATE] boolean := FALSE,
o  4822     jmv$swap_file_allocation_size: [XDCL, #GATE] 0 .. 0ffffffff(16) := 262144,
o  4823     lov$enable_source_type_checking: [XDCL, #GATE] boolean := FALSE,
o  4824     lov$ignore_param_verification: [XDCL, #GATE] integer := 0,
o  4825
o  4826 {  Debug, define maximum memory that 180 will use.  This is a way of limiting the amount of memory NOS/VE will
o  4827 {  use.  Can limit the memory less than defined upper bound.  Default to a value larger than existing memory
o  4828 {  sizes.
o  4829
o  4830     mmv$maximum_180_memory: [XDCL] integer := 1000000000000(16),
o  4831     mmv$shadow_by_segnum: [XDCL, #GATE] boolean := TRUE,
o  4832
o  4833 { Maximum segment length is no longer used by the OS. It must remain as a SETSA until release 1.6.1
o  4834 { because IM/DM uses it. As soon as they react to the change-this variable will be deleted.
o  4835
o  4836     mmv$max_segment_length: [XDCL, #GATE] integer := 2147483647,
o  4837     mtv$mx_ajl_entries: [XDCL, #GATE] 0 .. 0ffffffff(16) := 100,
o  4838     mtv$mx_segments: [XDCL] 0 .. 0ffffffff(16) := 0,
o  4839     mtv$mx_tasks: 0 .. 0ffffffff(16) := 255,  {this is a no-op and will be removed next release}
o  4840     mtv$scb_vector_sim_attribute: [XREF] 0 .. 255,
o  4841     osv$catalog_name_security: [XDCL, #GATE] boolean := FALSE,
o  4842     osv$default_pit: [XDCL] integer := 7fffffff(16), {default value for PIT}
```

System Constant Management
Global Declarations Declared by this module

```
o  4843      osv$delete_unreconciled_files: [XDCL, #GATE] boolean := FALSE,
o  4844      osv$dump_when_debug: [XDCL] boolean := FALSE,
o  4845      osv$emergency_intervention: [XDCL, #GATE] boolean := FALSE,
o  4846      osv$enable_hyperchannel: [XDCL, #GATE] boolean := FALSE,
o  4847      osv$reconcile_permanent_files: [XDCL, #GATE] boolean := FALSE,
o  4848      osv$recover_at_all_costs: [XDCL] boolean := FALSE,
o  4849      osv$reorganize_permanent_files: [XDCL, #GATE] boolean := FALSE,
o  4850      osv$validate_active_sets: [XDCL, #GATE] boolean := FALSE,
o  4851      osv$validate_permanent_files: [XDCL, #GATE] boolean := FALSE,
o  4852      pfv$restrict_catalog_flushing: [XDCL, #GATE] boolean := TRUE,
o  4853      osv$verify_missing_volumes: [XDCL, #GATE] boolean := TRUE,
o  4854      pfv$binary_catalog_search: [XDCL, #GATE] boolean := TRUE,
o  4855      pmv$debug_logging_enabled: [XDCL, #GATE] boolean := FALSE,
o  4856      pmv$constrain_meape_segments: [XDCL, #GATE] boolean := FALSE,
o  4857      rav$deadstart_intervention: [XDCL, #GATE] boolean := FALSE,
o  4858      rav$development_deadstart: [XDCL, #GATE] boolean := FALSE,
o  4859      rav$network_activation: [XDCL, #GATE] boolean := TRUE,
o  4860      rav$system_activation: [XDCL, #GATE] boolean := TRUE,
o  4861      syv$halt_on_exit_with_io: [XDCL, #GATE] boolean := FALSE,
o  4862      syv$nosve_internal_operations: [XDCL, #GATE] boolean := FALSE,
o  4863      nav$disable_network_relays: [XDCL, #GATE] boolean := FALSE,
o  4864      syv$user_templates: [XREF] boolean,
o  4865      tmv$display_actual_priority: [XDCL, #GATE] boolean := FALSE,
o  4866      dmv$pf_sparse: [XREF] boolean,
o  4867      syv$clone_enabled: [XREF] boolean,
o  4868      endvar: char;
o  4869
o  4870
o  4871  { NOTE:
o  4872  {   The following variable cannot be set via the set_system_attribute command.
o  4873  {   It is set via the change_secure_logging command during system operation.
o  4874
o  4875
o  4876    CONST
o  4877      clc$change_secure_logging_name = '* CHANGE_SECURE_LOGGING *';
o  4878
o  4879
o  4880    VAR
o  4881      clv$secure_logging_activated: [XDCL, #GATE] boolean := FALSE;
o  4882
o  4883
o  4884  { NOTE:
o  4885  {   The following variable cannot be set via the set_system_attribute command.
o  4886  {   It is set via the set_validation_level command during system operation.
o  4887
o  4888
o  4889    CONST
o  4890      avc$validation_level_const_name = '* VALIDATION_LEVEL *';
o  4891
o  4892
o  4893
o  4894
o  4895
o  4896
o  4897
```


System Constant Management
Global Declarations Declared by this module

```
o  4898    VAR
o  4899      avv$validation_level: [XDCL, #GATE] 0 .. 255 := 0;
o  4900
o  4901
o  4902
o  4903  { NOTE:
o  4904  {   The following variable cannot be set via the set_system_attribute command.
o  4905  {   It is set via the activate_system_logging and deactivate_system_logging
o  4906  {   commands during system operation.
o  4907
o  4908
o  4909    CONST
o  4910      clc$system_logging_active_name = '* SYSTEM_LOGGING_ACTIVE *';
o  4911
o  4912
o  4913    VAR
o  4914      clv$system_logging_activated: [XDCL, #GATE] boolean := FALSE;
o  4915
o  4916
o  4917  { Define values for the segment offset array.  This array defines page offsets
o  4918  { for the beginning of some special segments.  These segments do NOT start at
o  4919  { offset zero.  This is an attempt to reduce thrashing in the page map.  This
o  4920  { array is defined here because it may need to be model dependent.
o  4921
o  4922    VAR
o  4923      mmv$page_map_offsets: [XDCL, #GATE] mmt$page_map_offsets := [0, 0, 0, 0, 0];
o  4924
o  4925
o  4926    VAR
o  4927      dmv$maximum_allocation_size: [XREF] integer,
o  4928      dmv$quick_deadstart: [XREF] boolean,
o  4929      jmv$max_think_time: [XREF] integer,
o  4930      jmv$min_think_time: [XREF] integer,
o  4931      jsv$enable_swap_resident: [XREF] boolean,
o  4932      jsv$enable_swap_resident_no_io: [XREF] boolean,
o  4933      jsv$write_stale_pages: [XREF] boolean,
o  4934      mlv$wire_mli_tables: [XREF] boolean,
o  4935      mmv$advise_in_aio_limit: [XREF] integer,
o  4936      mmv$age_interval_ceiling: [XREF] 0 .. 255,
o  4937      mmv$age_interval_floor: [XREF] 0 .. 255,
o  4938      mmv$check_queues: [XREF] integer,
o  4939      mmv$maxws_aio_threshold: [XREF] integer,
o  4940      mmv$swapping_aic: [XREF] integer,
o  4941      osv$debug: [XREF] array [0..15] of integer,
o  4942      osv$default_sit_value: [XREF] integer,
o  4943      osv$disk_fault_simulation: [XREF] boolean,
o  4944      osv$error_idle_halt: [XREF] boolean,
o  4945      osv$keypoint_enable: [XREF] integer,
o  4946      osv$trap_task_errors: [XREF] boolean,
o  4947      sfv$dynamic_file_space_limits: [XREF] boolean,
o  4948      syv$allow_jr_test: [XREF] boolean,
o  4949      syv$debug_job_recovery: [XREF] boolean,
o  4950      syv$system_job_multiprocessing: [XREF] boolean,
o  4951      tmv$long_wait_force_swap_time: [XREF] integer,
o  4952      tmv$long_wait_swap_time: [XREF] integer;
```

System Constant Management
Global Declarations Declared by this module

o 4953

System Constant Management
Global Declarations Declared by this module
System Constant Definition Table

```
o  4955
o  4956    CONST
o  4957      max_name_length = 31,
o  4958      value_table_length = 179;
o  4959
o  4960 {  Note: The multiple shared queues feature introduces multiple shared queues and it also introduces the
o  4961 {        Manage Memory Utility.  The memory variables that it manages are also controlled by the
o  4962 {        SET_SYSTEM_ATTRIBUTE command.  So that users do not have to immediately change to use of the
o  4963 {        MMU, the old SETSA commands are being left in place.  For the attribute MINIMUM_SHARED_WORKING_SET
o  4964 {        the field controlled is to be mmv$gpql[mmc$pq_shared_task_service].minimum].  To make this
o  4965 {        change we need to change the appropriate entry in the value_table.  However, this can not be done
o  4966 {        because the object library generator will generate incorrect code.  A workaround has been coded.
o  4967 {        To remove the work-around, delete these comments, all lines referencing the
o  4968 {        VAR temp_workaround_min_sws and change the appropriate value_table entry.
o  4969
o  4970    VAR  temp_workaround_min_sws : [STATIC]  0..0ffffffff(16) := 0;
o  4971
o  4972
o  4973    VAR
o  4974      value_table: [READ, oss$mainframe_paged_literal] array [1 .. value_table_length] of value_table_type := [
o  4975 {    } [FALSE, 'ASSIGN_MULTIPLE_PAGES', 0, 100000, v_integer, ^mmv$assign_multiple_pages],
o  4976 {    } [FALSE,  avc$validation_level_const_name, 0, 2, v_byte, ^avv$validation_level],
o  4977 {    } [FALSE,  clc$change_secure_logging_name, 0, 1, v_boolean, ^clv$secure_logging_activated],
o  4978 {    } [FALSE,  clc$system_logging_active_name, 0, 1, v_boolean, ^clv$system_logging_activated],
o  4979 {    } [FALSE, 'ABORTED_TASK_THRESHOLD', 0, 7FFFFFFFFFFF(16), v_integer, ^mtv$aborted_task_threshold],
o  4980 {    } [FALSE, 'ADVISE_IN_LIMIT', 1, 1000, v_integer, ^mmv$advise_in_aio_limit],
o  4981 {    } [FALSE, 'AGE_INTERVAL_CEILING', 1, 255, v_byte, ^mmv$age_interval_ceiling],
o  4982 {    } [FALSE, 'AGE_INTERVAL_FLOOR', 1, 255, v_byte, ^mmv$age_interval_floor],
o  4983 {    } [FALSE, 'AGGRESSIVE_AGING_LEVEL', 0, 100000, v_integer, ^mmv$aggressive_aging_level],
o  4984 {    } [FALSE, 'AGGRESSIVE_AGING_LEVEL_2', 0, 100000, v_integer, ^mmv$aggressive_aging_level_2],
o  4985 {    } [FALSE, 'AGING_ALGORITHM', 0, 10000, v_integer, ^mmv$aging_algorithm],
o  4986 {    } [FALSE, 'AIO_LIMIT', 0, 5000000, v_integer, ^mmv$aio_limit],
o  4987 {    } [FALSE, 'ALLOW_JR_TEST', 0, 1, v_boolean, ^syv$allow_jr_test],
o  4988 {    } [FALSE, 'AUTOMATIC_UNSTEP_RESUME', 0, 1, v_boolean, ^mtv$automatic_unstep_resume],
o  4989 {    } [TRUE , 'CATALOG_NAME_SECURITY', 0, 1, v_boolean, ^osv$catalog_name_security],
o  4990 {    } [FALSE, 'CHECK_IDLE_DISPATCHING_INTERVAL', 1000, 1000000000, v_integer,
o  4991                ^jmv$scan_idle_dispatch_interval],
o  4992 { -------- Declarations for forcing the use of cache and maps omitted at compile time --------
o  4993 {    } [FALSE, 'CLONE_ENABLED', 0, 1, v_boolean, ^syv$clone_enabled],
o  4994 {    } [FALSE, 'COMMAND_STATISTICS_ENABLED', 0, 1, v_boolean, ^clv$command_statistics_enabled],
o  4995 {    } [FALSE, 'CYCLE_WAIT_TIME', 0, 100000000, v_integer, ^tmv$cycle_delay_time],
o  4996 {    } [TRUE,  'DEADSTART_INTERVENTION', 0, 1, v_boolean, ^rav$deadstart_intervention],
o  4997 {    } [FALSE, 'DEBUG0', 0, 0ffffffffffff(16), v_integer, ^osv$debug[0]],
o  4998 {    } [FALSE, 'DEBUG1', 0, 0ffffffffffff(16), v_integer, ^osv$debug[1]],
o  4999 {    } [FALSE, 'DEBUG2', 0, 0ffffffffffff(16), v_integer, ^osv$debug[2]],
o  5000 {    } [FALSE, 'DEBUG3', 0, 0ffffffffffff(16), v_integer, ^osv$debug[3]],
o  5001 {    } [FALSE, 'DEBUG4', 0, 0ffffffffffff(16), v_integer, ^osv$debug[4]],
o  5002 {    } [FALSE, 'DEBUG5', 0, 0ffffffffffff(16), v_integer, ^osv$debug[5]],
o  5003 {    } [FALSE, 'DEBUG6', 0, 0ffffffffffff(16), v_integer, ^osv$debug[6]],
o  5004 {    } [FALSE, 'DEBUG7', 0, 0ffffffffffff(16), v_integer, ^osv$debug[7]],
o  5005 {    } [FALSE, 'DEBUG8', 0, 0ffffffffffff(16), v_integer, ^osv$debug[8]],
o  5006 {    } [FALSE, 'DEBUG9', 0, 0ffffffffffff(16), v_integer, ^osv$debug[9]],
o  5007 {    } [FALSE, 'DEBUG10', 0, 0ffffffffffff(16), v_integer, ^osv$debug[10]],
o  5008 {    } [FALSE, 'DEBUG11', 0, 0ffffffffffff(16), v_integer, ^osv$debug[11]],
```

System Constant Management
  Global Declarations Declared by this module
    System Constant Definition Table

```
o  5009 {    } [FALSE, 'DEBUG12', 0, 0ffffffffffff[16], v_integer, ^osv$debug[12]],
o  5010 {    } [FALSE, 'DEBUG13', 0, 0ffffffffffff[16], v_integer, ^osv$debug[13]],
o  5011 {    } [FALSE, 'DEBUG14', 0, 0ffffffffffff[16], v_integer, ^osv$debug[14]],
o  5012 {    } [FALSE, 'DEBUG15', 0, 0ffffffffffff[16], v_integer, ^osv$debug[15]],
o  5013 {    } [FALSE, 'DEBUG_JOB_RECOVERY', 0, 1, v_boolean, ^syv$debug_job_recovery],
o  5014 {    } [FALSE, 'DEDICATE_A_CPU_TO_NOS', 0, 1, v_boolean, ^tmv$dedicate_a_cpu_to_nos],
o  5015 {    } [FALSE, 'DEFAULT_PIT', 1, 0ffffffff[16], v_integer, ^osv$default_pit],
o  5016 {    } [FALSE, 'DEFAULT_SIT', 500, 1000000000, v_integer, ^osv$default_sit_value],
o  5017 {    } [TRUE,  'DELETE_UNRECONCILED_FILES', 0, 1, v_boolean, ^osv$delete_unreconciled_files],
o  5018 {    } [TRUE,  'DEVELOPMENT_DEADSTART', 0, 1, v_boolean, ^rav$development_deadstart],
o  5019 {    } [FALSE, 'DISABLE_WRITE_FOR_PERF_MEAS', 0, 1, v_boolean, ^mmv$disable_write_for_perf_meas],
o  5020 {    } [FALSE, 'DISDELAY', 1, 100000, v_integer, ^dpv$display_delay],
o  5021 {    } [TRUE,  'DISPLAY_DEADSTART_MESSAGES', 0, 1, v_boolean, ^dsv$display_deadstart_messages ],
o  5022 {    } [FALSE, 'DUMP_WHEN_DEBUG', 0, 1, v_boolean, ^osv$dump_when_debug],
o  5023 {    } [FALSE, 'DYNAMIC_FILE_SPACE_LIMITS', 0, 1, v_boolean, ^sfv$dynamic_file_space_limits],
o  5024 {    } [FALSE, 'ENABLE_DEBUG_CODE', 0, 1, v_boolean, ^jsv$enable_debug_code],
o  5025 {    } [FALSE, 'ENABLE_FAULT_INJECTION', 0, 1, v_boolean, ^syv$enable_fault_injection],
o  5026 {    } [TRUE ,  'ENABLE_HEAP_TRACE', 0, 1, v_boolean, ^syv$enable_heap_trace],
o  5027 {    } [TRUE,  'ENABLE_HYPERCHANNEL', 0, 1, v_boolean, ^osv$enable_hyperchannel],
o  5028 {    } [FALSE, 'ENABLE_QUEUE_FILE_ACCESS', 0, 1, v_boolean, ^jmv$enable_queue_file_access],
o  5029 {    } [FALSE, 'ENABLE_SOURCE_CYBIL_CHECKING', 0, 1, v_boolean, ^lov$enable_source_type_checking],
o  5030 {    } [FALSE, 'ENABLE_SWAP_FILE_STATISTICS', 0, 1, v_boolean, ^jsv$enable_swap_file_statistics],
o  5031 {    } [FALSE, 'ENABLE_SWAP_RESIDENT', 0, 1, v_boolean, ^jsv$enable_swap_resident],
o  5032 {    } [FALSE, 'ENABLE_SWAP_RESIDENT_NO_IO', 0, 1, v_boolean, ^jsv$enable_swap_resident_no_io],
o  5033 {    } [FALSE, 'FILE_ALLOCATION_INTERVAL', 0, 7fffffff[16], v_integer, ^mmv$file_allocation_interval],
o  5034 {    } [FALSE, 'FILE_SERVER_DEBUG_ENABLED', 0, 1, v_boolean, ^dfv$file_server_debug_enabled],
o  5035 {    } [FALSE, 'FILE_SERVER_RECOVERY_ENABLED', 0, 1, v_boolean, ^dfv$job_recovery_enabled],
o  5036 {    } [FALSE, 'FILE_SERVER_INFO_ENABLED', 0, 1, v_boolean, ^dfv$file_server_info_enabled],
o  5037 {    } [FALSE, 'FORCE_DIRECT_TAPE_IO', 0, 1, v_boolean, ^bav$force_direct_tape_io ],
o  5038 {    } [FALSE, 'FREE_WORKING_SET_ON_SWAPOUT', 0, 1, v_boolean, ^jsv$free_working_set_on_swapout],
o  5039 {    } [FALSE, 'HALTRING', 0, 16, v_byte, ^mtv$halt_cpu_ring_number],
o  5040 {    } [FALSE, 'HALT_ON_HUNG_TASK', 0, 1, v_boolean, ^tmv$halt_on_hung_task],
o  5041 {    } [FALSE, 'HALT_ON_PROCESSOR_MALFUNCTION', 0, 1, v_boolean, ^mtv$halt_on_proc_malf],
o  5042 {    } [FALSE, 'HALT_ON_SWAPIN_FAILURE', 0, 1, v_boolean, ^jsv$halt_on_swapin_failure],
o  5043 {    } [FALSE, 'IF_CONDITION_HANDLING_TRACE', 0, 1, v_boolean, ^iiv$condition_handler_trace],
o  5044 {    } [FALSE, 'IGNORE_IMAGE', 0, 1, v_boolean, ^dsv$ignore_image],
o  5045 {    } [FALSE, 'IGNORE_PARAMETER_VERIFICATION', 0, 2, v_integer, ^lov$ignore_param_verification],
o  5046 {    } [FALSE, 'TRIM_FILES', 0, 1, v_boolean, ^dmv$trim_files],
o  5047 {    } [TRUE,  'UNLOAD_DEADSTART_TAPE', 0, 1, v_boolean, ^dsv$unload_deadstart_tape],
o  5048 {    } [TRUE ,  'JOB_RECOVERY_OPTION', 0, 100, v_integer, ^syv$job_recovery_option],
o  5049 {    } [FALSE, 'JOB_WORKING_SET_AGE_INTERVAL', 1000, 7fffffff[16], v_integer, ^mmv$jws_queue_age_interval],
o  5050 {    } [FALSE, 'KCU_ENABLE', 0, 100, v_integer, ^osv$keypoint_enable],
o  5051 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5052 {    } [FALSE, 'LONG_WAIT_SWAP_TIME', 0, 1000000000, v_integer, ^tmv$long_wait_swap_time],
o  5053 {    } [FALSE, 'LONG_WAIT_FORCE_SWAP_TIME', 0, 1000000000, v_integer, ^tmv$long_wait_force_swap_time],
o  5054 {    } [FALSE, 'MANDATORY_DUALSTATE', 0, 1, v_boolean, ^syv$mandatory_dualstate],
o  5055 {    } [TRUE,  'MAXIMUM_180_MEMORY', 1000000, 400000000, v_integer, ^mmv$maximum_180_memory],
o  5056 {    } [TRUE,  'MAXIMUM_ACTIVE_SEGMENTS', 100, 65535, v_halfword, ^mtv$mx_segments],
o  5057 {    } [FALSE, 'MAXIMUM_BYTES_PER_TAPE_IO', 1, 7FFFFFFF[16], v_integer, ^bav$max_bytes_per_tape_io ],
o  5058 {    } [FALSE, 'MAXIMUM_INDIRECT_TAPE_BLOCK', 1, 7FFFFFFF[16], v_integer, ^bav$max_indirect_tape_block ],
o  5059 {    } [FALSE, 'MAXIMUM_KNOWN_JOBS', 1, jmc$maximum_job_count, v_halfword, ^jmv$maximum_known_jobs],
o  5060 {    } [FALSE, 'MAXIMUM_OUTPUT_FILES', 1, jmc$maximum_output_count, v_halfword, ^jmv$maximum_known_outputs],
o  5061 {    } [FALSE, 'MAXIMUM_PAGES_NO_FILE', 0, 7fffffff[16], v_integer, ^mmv$max_pages_no_file],
o  5062 {    } [FALSE, 'MAXIMUM_PAGES_TO_SWAP', 0, 0ffff[16], v_integer, ^jsv$maximum_pages_to_swap],
```


System Constant Management
  Global Declarations Declared by this module
    System Constant Definition Table

```
o  5063 {    } [FALSE, 'MAX_PAGES_FIRST_SWAP_TASK', 0, 0ffff[16], v_integer, ^jsv$max_pages_first_swap_task],
o  5064
o  5065 { Maximum segment length is no longer used by the OS. It must remain as a SETSA until release 1.6.1
o  5066 { because IM/DM uses it. As soon as they react to the change-this variable will be deleted.
o  5067
o  5068 {    } [FALSE, 'MAXIMUM_SEGMENT_LENGTH', 150000000, 7fffffff[16], v_integer, ^mmv$max_segment_length],
o  5069 {    } [FALSE, 'MAXIMUM_SWAP_RESIDENT_TIME', 0, 86400000000, v_integer, ^jsv$max_time_swap_io_complete],
o  5070 {    } [FALSE, 'MAXIMUM_TAPE_BLOCK_SIZE', 1, 7FFFFFFF[16], v_integer, ^bav$max_allowed_tape_block_size ],
o  5071 {    } [TRUE,  'MAXIMUM_TASKS', 20, osc$max_tasks, v_halfword, ^mtv$mx_tasks],
o  5072 {    } [FALSE, 'MAXIMUM_THINK_TIME', 0, 1000000000, v_integer, ^jmv$max_think_time],
o  5073 {    } [FALSE, 'MAXWS_AIO_THRESHOLD', 0, 100000, v_integer, ^mmv$maxws_aio_threshold],
o  5074 {    } [FALSE, 'MAX_TIME_SWAP_IO_NOT_INIT', 0, 86400000000, v_integer, ^jsv$max_time_swap_io_not_init],
o  5075 {    } [FALSE, 'MINIMUM_AVAILABLE_PAGES', 0, 100000, v_integer, ^mmv$min_avail_pages],
o  5076 {    } [FALSE, 'MINIMUM_SHARED_WORKING_SET', 0, 100000, v_halfword,
o  5077              ^temp_workaround_min_sws],
o  5078 {  Note: The following line should be made active when the temp_workaround is no longer needed
o  5079 {            ^mmv$gpql[mmc$pq_shared_task_service].minimum],
o  5080
o  5081 {    } [FALSE, 'MINIMUM_THINK_TIME', 0, 1000000000, v_integer, ^jmv$min_think_time],
o  5082 {    } [FALSE, 'MM_CHECK_QUEUES', 0, 2, v_integer, ^mmv$check_queues],
o  5083 {    } [FALSE, 'MM_PERIODIC_CALL', 1000, 10000000, v_integer, ^mmv$periodic_call_interval],
o  5084 {    } [FALSE, 'MULTI_PAGE_WRITE', 0, 1, v_boolean, ^mmv$multi_page_write],
o  5085 {    } [TRUE,  'NETWORK_ACTIVATION', 0, 1, v_boolean, ^rav$network_activation],
o  5086 {    } [FALSE, 'NOSVE_INTERNAL_OPERATIONS', 0, 1, v_boolean, ^syv$nosve_internal_operations],
o  5087 {    } [FALSE, 'NO_MEMORY_BUFFERING', 0, 1, v_boolean, ^mmv$no_memory_buffering],
o  5088 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5089 {    } [FALSE, 'OUTPUT_OPTION', 0, 100000000, v_integer, ^iiv$output_option],
o  5090 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5091 {    } [FALSE, 'PERMANENT_FILE_OVERFLOW', 0, 1, v_boolean, ^dmv$permanent_file_overflow],
o  5092 {    } [FALSE, 'PROCESSOR_DUE_THRESHOLD', 0, 7FFFFFFFFFFF[16], v_integer, ^mtv$processor_due_threshold],
o  5093 {    } [FALSE, 'QUICKDS', 0, 1, v_boolean, ^dmv$quick_deadstart],
o  5094 {    } [FALSE, 'READ_TU_EXECUTE', 1, 32, v_halfword, ^mmv$read_tu_execute],
o  5095 {    } [FALSE, 'READ_TU_READ_WRITE', 1, 32, v_halfword, ^mmv$read_tu_read_write],
o  5096 {    } [TRUE,  'RECONCILE_PERMANENT_FILES', 0, 1, v_boolean, ^osv$reconcile_permanent_files],
o  5097 {    } [TRUE,  'RECOVER_AT_ALL_COSTS', 0, 1, v_boolean, ^osv$recover_at_all_costs],
o  5098 {    } [FALSE, 'RECYCLE_DEVICE_LOG', 0, 1, v_boolean, ^dmv$recycle_device_log],
o  5099 {    } [TRUE,  'REORGANIZE_PERMANENT_FILES', 0, 1, v_boolean, ^osv$reorganize_permanent_files],
o  5100 {    } [FALSE, 'SCHEDULER_WAIT_TIME', 0, 5000000, v_integer, ^jmv$scheduler_wait_time],
o  5101 {    } [FALSE, 'SCHED_MEMORY_WAIT_FACTOR', 0, 100, v_integer, ^jmv$sched_memory_wait_factor],
o  5102 {    } [FALSE, 'SHARED_PAGES_IN_JWS', 0, 1, v_boolean, ^mmv$shared_pages_in_jws],
o  5103 {    } [FALSE, 'SHARED_WORKING_SET_AGE_INTERVAL', 0, 7fffffff[16], v_integer, ^mmv$shared_queue_age_interval],
o  5104 {    } [FALSE, 'SWAPPING_AIC', 0, 100, v_integer, ^mmv$swapping_aic],
o  5105 {    } [FALSE, 'SWAP_FILE_ALLOCATION_SIZE', 4096, 1000000, v_halfword, ^jmv$swap_file_allocation_size],
o  5106 {    } [FALSE, 'SWAP_JOBS_IN_LONG_WAIT', 0, 1, v_boolean, ^jmv$swap_jobs_in_long_wait],
o  5107 {    } [FALSE, 'SYSTEM_DEBUG_RING', 0, 15, v_integer, ^tmv$system_debug_ring],
o  5108 {    } [FALSE, 'SYSTEM_DEBUG_SEGMENT', 0, 4095, v_integer, ^tmv$system_debug_segment],
o  5109 {    } [FALSE, 'SYSTEM_ERROR_HANG_COUNT', 0, 1000, v_halfword, ^tmv$system_error_hang_count],
o  5110 {    } [FALSE, 'SYSTEM_HALTRING', 0, 16, v_byte, ^mtv$system_haltring],
o  5111 {    } [TRUE,  'SYSTEM_JOB_MULTIPROCESSING', 0, 1, v_boolean, ^syv$system_job_multiprocessing],
o  5112 {    } [FALSE, 'TEMPORARY_FILE_OVERFLOW', 0, 1, v_boolean, ^dmv$temporary_file_overflow],
o  5113 {    } [FALSE, 'TEST_RECOVERY', 0, 1, v_boolean, ^dmv$test_recovery],
o  5114 {    } [FALSE, 'THINK_EXPIRATION_TIME', 0, 1000000000, v_integer, ^jsv$think_expiration_time],
o  5115 {    } [FALSE, 'USE_ASSIGN_PAGES_FOR_TAPE', 0, 1, v_boolean, ^bav$use_assign_pages_for_tape ],
o  5116 {    } [TRUE,  'VALIDATE_PERMANENT_FILES', 0, 1, v_boolean, ^osv$validate_permanent_files],
```

System Constant Management
  Global Declarations Declared by this module
    System Constant Definition Table

```
o  5117 {    } [TRUE,  'VALIDATE_SYSTEM_SET', 0, 1, v_boolean, ^osv$validate_active_sets],
o  5118 {    } [FALSE, 'VECTOR_SIMULATION', 0, 2, v_byte, ^mtv$scb_vector_sim_attribute],
o  5119 {    } [FALSE, 'VERIFY_HEAP_LINKAGE', 0, 1, v_boolean, ^syv$verify_heap_linkage],
o  5120 {    } [TRUE,  'VOLUME_CLASS', 0, 1, v_boolean, ^dmv$volume_class_kludge],
o  5121 {    } [TRUE,  'WIRE_MLI_TABLES', 0, 1, v_boolean, ^mlv$wire_mli_tables],
o  5122 {    } [FALSE, 'WRITE_AGED_OUT_PAGES', 0, 100000, v_integer, ^mmv$write_aged_out_pages],
o  5123 {    } [TRUE,  'WRITE_STALE_PAGES',0, 1, v_boolean, ^jsv$write_stale_pages],
o  5124 {    } [TRUE,  'MAXIMUM_ACTIVE_JOBS', 1, jmc$max_active_jobs, v_halfword, ^mtv$mx_ajl_entries],
o  5125 {    } [false,                  'unused_boolean_for_io', 0, 1, v_boolean, ^iov$unused_boolean],
o  5126 {    } [FALSE, 'MAXIMUM_JOB_CLASSES', jmc$minimum_job_classes, jmc$maximum_job_classes, v_halfword,
o  5127              ^jmv$maximum_job_classes],
o  5128 {    } [FALSE, 'MAXIMUM_SERVICE_CLASSES', jmc$minimum_service_classes, jmc$maximum_service_classes, v_halfword,
o  5129              ^jmv$maximum_service_classes],
o  5130 {    } [FALSE, 'MAXIMUM_DEBUG_OUTPUT_LINES', 100, 7FFFFFFFFFFF(16), v_integer, ^syv$max_debug_output_lines],
o  5131 {    } [FALSE, 'DISK_FAULT_SIMULATION', 0, 1, v_boolean, ^osv$disk_fault_simulation],
o  5132 {    } [FALSE, 'EMERGENCY_INTERVENTION', 0, 1, v_boolean, ^osv$emergency_intervention],
o  5133 {    } [FALSE, 'DISABLE_NETWORK_RELAYS', 0, 1, v_boolean, ^nav$disable_network_relays],
o  5134 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5135 {    } [FALSE, 'SHADOW_BY_SEGNUM', 0, 1, v_boolean, ^mmv$shadow_by_segnum],
o  5136 {    } [FALSE, 'ENABLE_PM_DEBUG_LOGGING', 0, 1, v_boolean, ^pmv$debug_logging_enabled],
o  5137 {    } [FALSE, 'CONSTRAIN_MEAPE_SEGMENTS', 0, 1, v_boolean, ^pmv$constrain_meape_segments],
o  5138 {    } [TRUE,  'SYSTEM_ACTIVATION', 0, 1, v_boolean, ^rav$system_activation],
o  5139 {    } [FALSE, 'USER_TEMPLATES', 0, 1, v_boolean, ^syv$user_templates],
o  5140 {    } [FALSE, 'CLIENT_JOB_LISTS', 1, dfc$max_job_list_p_array_size, v_integer,
o  5141              ^dfv$maximum_client_job_lists],
o  5142 {    } [FALSE, 'SERVED_FAMILY_LISTS', 1, dfc$maximum_family_lists, v_integer,
o  5143         ^dfv$number_served_family_lists],
o  5144 {    } [FALSE, 'SPECIAL_TRAP', 0, 1, v_boolean, ^osv$special_aam_trap],
o  5145 {    } [FALSE, 'TRAP_TASK_ERRORS', 0, 1, v_boolean, ^osv$trap_task_errors],
o  5146 {    } [FALSE, 'DISPLAY_ACTUAL_PRIORITY', 0, 1, v_boolean, ^tmv$display_actual_priority],
o  5147 {    } [FALSE, 'TIMED_WAIT_NOT_QUEUED', 60000000, 7FFFFFFFFFF(16), v_integer, ^tmv$timed_wait_not_queued],
o  5148 {    } [TRUE,  'VALIDATE_ACTIVE_SETS', 0, 1, v_boolean, ^osv$validate_active_sets],
o  5149 {    } [FALSE, 'PF_SPARSE', 0, 1, v_boolean, ^dmv$pf_sparse],
o  5150 {    } [FALSE, 'MAXIMUM_ALLOCATION_SIZE', 16384, 0FFFFFFFFFFFF(16), v_integer, ^dmv$maximum_allocation_size],
o  5151 {    } [FALSE, 'DEBUGGER_PAGE_WAIT_LINES', 0, 07FFFFFFFFFFFFFF(16), v_integer, ^syv$debugger_page_wait_lines],
o  5152 {    } [FALSE, 'SPACE_MESSAGES_TO_CONSOLE', 0, 1, v_boolean, ^dmv$space_messages_to_console],
o  5153 {    } [FALSE, 'DEFAULT_DEBUG_OUTPUT_DISPOSAL', 0, 3, v_byte, ^syv$dflt_debug_output_disposal.byte],
o  5154 {    } [TRUE,  'ENABLE_CONSOLE_BELL', 0, 1, v_boolean, ^dpv$enable_console_bell],
o  5155 {    } [FALSE, 'BINARY_CATALOG_SEARCH', 0, 1, v_boolean, ^pfv$binary_catalog_search],
o  5156 {    } [FALSE, 'HALT_ON_EXIT_WITH_IO', 0, 1, v_boolean, ^syv$halt_on_exit_with_io],
o  5157 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5158 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5159 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5160 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5161 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5162 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5163 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5164 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5165 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5166 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL],
o  5167 {    } [FALSE, 'dummy', 0, 1, v_boolean, NIL]];
o  5168
o  5169
```


System Constant Management
  [XDCL] syp$fetch_system_const_from_ssr

```
o  5172
o  5173   PROCEDURE [XDCL] syp$fetch_system_const_from_ssr
4  5174     (VAR status: ost$status);
4  5175
4  5176     status.normal := TRUE;
4  5177   PROCEND syp$fetch_system_const_from_ssr;
o  5178
```

System Constant Management
[XDCL, #GATE] syp$fetch_system_constant

```
       0  5180
       0  5181    PROCEDURE [XDCL, #GATE] syp$fetch_system_constant
       0  5182      (VAR name: string ( * );
       0  5183       VAR index: integer;
       0  5184       VAR value: integer;
       0  5185       VAR status: ost$status);
       0  5186
       0  5187      VAR
       0  5188        tab_p: ^value_table_type,
       0  5189        temp_status: ost$status;
       0  5190
       0  5191      temp_workaround_min_sws := mmv$gpql[mmc$pq_shared_task_service].minimum;
       4  5192      IF name = 'EVERYTHING' THEN
      68  5193         IF (index >= 1) AND (index <= value_table_length) THEN
      7E  5194             name := value_table [index].name;
      D6  5195             IF name <> 'dummy' THEN
     120  5196                CASE value_table [index].value_type OF
     144  5197                = v_integer =
     144  5198                  value := value_table [index].integer_p^;
     162  5199                = v_halfword =
     162  5200                  value := value_table [index].halfword_p^;
     17A  5201                = v_byte =
     17A  5202                  value := value_table [index].byte_p^;
     192  5203                = v_boolean =
     192  5204                  value := ORD (value_table [index].boolean_p^);
     1A4  5205                CASEND;
     1A4  5206             IFEND;
     1A4  5207             IF index = value_table_length THEN
     1B0  5208                index := 0;
     1BA  5209             ELSE
     1BA  5210                index := index + 1;
     1C0  5211             IFEND;
     1C0  5212         IFEND;
     1C2  5213      ELSE
*WARN*   5214         search_value_table (name, tab_p, status);
     232  5215         IF status.normal THEN
     23C  5216           CASE tab_p^.value_type OF
     25A  5217           = v_integer =
     25A  5218             value := tab_p^.integer_p^;
     270  5219           = v_halfword =
     270  5220             value := tab_p^.halfword_p^;
     282  5221           = v_byte =
     282  5222             value := tab_p^.byte_p^;
     294  5223           = v_boolean =
     294  5224             value := ORD (tab_p^.boolean_p^);
     2A4  5225           CASEND;
     2A4  5226           RETURN;
     2A6  5227         IFEND;
     2A6  5228
     2A6  5229 { The following code is executed only if the status returned from search_value_table was abnormal.
     2A6  5230 { Use a temporary status in the following procedure calls to preserve the bad status from search_value_table.
     2A6  5231
     2A6  5232         dmp$fetch_debug_option_value (name, value, temp_status);
     312  5233         IF temp_status.normal THEN
     31A  5234            RETURN;
```


System Constant Management
[XDCL, #GATE] syp$fetch_system_constant

```
     31C  5235         IFEND;
     31C  5236
     31C  5237         iop$fetch_debug_option_value (name, value, temp_status);
     382  5238       IFEND;
     382  5239    PROCEND syp$fetch_system_constant;
       0  5240
```

System Constant Management
[XDCL] syp$save_system_const_in_ssr

```
0  5242
0  5243   PROCEDURE [XDCL] syp$save_system_const_in_ssr
4  5244     (VAR status: ost$status);
4  5245
4  5246     status.normal := TRUE;
4  5247   PROCEND syp$save_system_const_in_ssr;
0  5248
```


System Constant Management
[XDCL] syp$set_processor_attributes

```
0  5250
0  5251   PROCEDURE [XDCL] syp$set_processor_attributes
0  5252     (    model_number: integer;
0  5253      VAR status: ost$status);
0  5254
0  5255     VAR
0  5256       p: ^RECORD
0  5257         fill1: string (12),
0  5258         a0_offset: 0 .. 0ffffffff(16),
0  5259         fill2: string (4),
0  5260         a1_offset: 0 .. 0ffffffff(16),
0  5261       RECEND;
0  5262
0  5263     pmv$quantum := osv$global_processor_model_info.quantum;
4  5264     mmv$tick_time := osv$global_processor_model_info.tick_time;
4  5265
4  5266 { The default values of the variables managed by the Manage Memory Utility have already been saved in the
4  5267 { procedure mmp$initialize in mmm$deadstart_initialization.  But now mmv$tick_time has been reset.
4  5268 { Therefore reset the default in mmv$manage_memory_utility
4  5269
4  5270     mmv$manage_memory_utility.ma[ma_tt].default := mmv$tick_time;
4  5271
4  5272     mtv$xp_initial_value.tos_registers[1].pva.offset :=
4  5273          mmv$page_map_offsets [mmc$pmo_r1_stack] * osv$page_size +
4  5274               mmc$ring_crossing_offset;
4  5275     mtv$xp_initial_value.tos_registers[2].pva.offset :=
4  5276          mmv$page_map_offsets [mmc$pmo_r2_stack] * osv$page_size +
4  5277               mmc$ring_crossing_offset;
4  5278     mtv$xp_initial_value.tos_registers[3].pva.offset :=
4  5279          mmv$page_map_offsets [mmc$pmo_r3_stack] * osv$page_size +
4  5280               mmc$ring_crossing_offset;
4  5281     p := #LOC (mtv$xp_initial_value);
62 5282     p^.a0_offset := mmv$page_map_offsets [mmc$pmo_r1_stack] * osv$page_size +
62 5283               mmc$ring_crossing_offset;
62 5284     p^.a1_offset := mmv$page_map_offsets [mmc$pmo_r1_stack] * osv$page_size +
62 5285               mmc$ring_crossing_offset;
62 5286   PROCEND syp$set_processor_attributes;
0  5287
```

System Constant Management
[XDCL, #GATE] syp$store_system_constant

```
      0  5289
      0  5290    PROCEDURE [XDCL, #GATE] syp$store_system_constant
      0  5291      (     name: string ( * );
      0  5292            index: integer;
      0  5293            value: integer;
      0  5294       VAR status: ost$status);
      0  5295
      0  5296      VAR
      0  5297        tab_p: ^value_table_type,
      0  5298        temp_status: ost$status;
      0  5299
      0  5300      search_value_table (name, tab_p, status);
     22  5301      IF status.normal THEN
     2A  5302
     2A  5303        IF (mtv$sys_core_init_complete) AND (tab_p^.deadstart_only) THEN
     42  5304          osp$set_status_abnormal ('SY', sye$not_changeable_after_ds,
     76  5305            'Variable not changeable after ds complete', status);
     76  5306          RETURN;
     78  5307        IFEND;
     78  5308
     78  5309        IF (value > tab_p^.max) OR (value < tab_p^.min) THEN
     90  5310          osp$set_status_abnormal ('SY', sye$range_error, 'Value out of range', status);
     C6  5311        ELSE
     C6  5312          temp_workaround_min_sws := mmv$gpq1[mmc$pq_shared_task_service].minimum;
     C6  5313          CASE tab_p^.value_type OF
     F0  5314          = v_integer =
     F0  5315            tab_p^.integer_p^ := value;
    104  5316          = v_halfword =
    104  5317            tab_p^.halfword_p^ := value;
    110  5318          = v_byte =
    110  5319            tab_p^.byte_p^ := value;
    11C  5320          = v_boolean =
    11C  5321            IF value = ORD (FALSE) THEN
    120  5322              tab_p^.boolean_p^ := FALSE;
    12E  5323            ELSE
    12E  5324              tab_p^.boolean_p^ := TRUE;
    136  5325            IFEND;
    136  5326          CASEND;
    136  5327          mmv$gpq1[mmc$pq_shared_task_service].minimum := temp_workaround_min_sws;
    13E  5328        IFEND;
    13E  5329        RETURN;
    140  5330      IFEND;
    140  5331
    140  5332      osp$set_status_abnormal ('OS', cle$name_not_a_keyword_value, name, status);
    16C  5333      osp$append_status_parameter (osc$status_parameter_delimiter, ' for parameter NAME', status);
    194  5334
    194  5335  { Use a temporary status in the following procedure calls to preserve the status returned for
    194  5336  { search_value_table.
    194  5337
    194  5338      dmp$store_debug_option_value (name, value, temp_status);
    1B4  5339      IF temp_status.normal THEN
    1BC  5340        RETURN;
    1BE  5341      IFEND;
    1BE  5342
    1BE  5343      iop$store_debug_option_value (name, value, temp_status);
```


System Constant Management
[XDCL, #GATE] syp$store_system_constant

```
    1E2  5344
    1E2  5345
    1E2  5346    PROCEND syp$store_system_constant;
      0  5347
```

System Constant Management
search_value_table

```
      0  5349
      0  5350   PROCEDURE search_value_table
      0  5351     (    name: string ( * );
      0  5352      VAR tab_p: ^value_table_type;
      0  5353      VAR status: ost$status);
      0  5354
      0  5355     VAR
      0  5356       error_string: string (55),
      0  5357       i: integer,
      0  5358       lname: string (max_name_length),
      0  5359       table_i: integer;
      0  5360
      0  5361     table_i := 1;
      4  5362     lname := name;
     26  5363     FOR i := 1 TO STRLENGTH (lname) DO
     30  5364       IF (lname (i) >= 'a') AND (lname (i) <= 'z') THEN
     48  5365         lname (i) := CHR (ORD (lname (i)) - ORD ('a') + ORD ('A'));
     50  5366       IFEND;
     50  5367     FOREND;
     54  5368
     54  5369   /lp/
     54  5370     WHILE lname <> value_table [table_i].name DO
     74  5371       table_i := table_i + 1;
     74  5372       IF table_i <= value_table_length THEN
     7E  5373         CYCLE /lp/
     82  5374       IFEND;
     82  5375       error_string := 'Unknown parameter name: ';
     90  5376       error_string(25, *) := lname;
     9A  5377       osp$set_status_abnormal ('SY', sye$unknown_parameter_name, lname, status);
     CA  5378       RETURN;
     CC  5379     WHILEND /lp/;
     E8  5380
     E8  5381     tab_p := ^value_table [table_i];
     E8  5382   PROCEND search_value_table;
      0  5383


      0  5385 MODEND sym$system_constant_manager;
```

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

System Constant Management
search_value_table

```
         ERROR     LINE  TEXT
WARNING  CY 821    5214   Code scheduling abandoned for this block due to register jamming.
```

LEVEL SUMMARY
**** 1 warning diagnostic

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
a0_offset	5258	5282/M							
a1_offset	5260	5284/M							
amc$file_byte_limit	2938	2941	2943						
amt$file_byte_address	2941	2758	2759						
amt$file_length	2943	2807							
avc$validation_level_const_name	4890	4976							
avt$account_name	1854	1822	2357	2362	2373	2379	3020		
avt$project_name	1858	1823	2363	2380	3021				
avv$validation_level	4899	4976							
bav$force_direct_tape_io	1722	5037							
bav$max_allowed_tape_block_size	1724	5070							
bav$max_bytes_per_tape_io	1726	5057							
bav$max_indirect_tape_block	1728	5058							
bav$use_assign_pages_for_tape	1730	5115							
boolean_p	4803	5204	5224	5322/M	5324/M				
byte	4719	5153							
byte_p	4801	5202	5222	5319/M					
clc$change_secure_logging_name	4877	4977							
clc$min_ecc	15	16	28	30	33	36	39	42	45
		48	60	64	67	72	77	80	85
		88	93	96	101	106	109	113	119
		122	125	128	133	136	143	146	149
		154	157	162	165	169	172	177	183
		186	190	193	198	201	205	208	212
		217	220	225	230	235	240	243	246
		249	252	255	261	264	268	272	278
		281	284	287	290	295	298	301	307
		310	315	318	323	327	333	336	339
		342	347	352	355	358	361	365	368
		372	377	381	385	388	392	402	407
		412	416	422	426	431	435	438	441
		444	447	450	455	458	461	464	468
		473	476	482	485	489	492	495	498
		501	504	508	512	515	518	521	524
		527	530	533	537	540	543	546	549
		552	555	558	561	564	567	570	573
		576	580	583	586	589	593	597	601
		605	609	612	616	619	622	625	628
		631	635	638	643	648	652	656	659
		662	665	668	672	675	678	683	686
		690	694	698	704	707	710	713	716
		719	722	725	731	734	738	742	745
		750	753	758	763	768	773	776	779
		783	788	791	795	799	802	806	809
		812	815	820	823	827	831	835	839
		842	847	851	857	865	869	875	883
		886	891	894	898	902	906	910	916
		920	925	928	932	936	940	944	948
		953	956	960	964	970	973	976	979

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		982	985	988	993	996	1000	1005	1009
		1012	1015	1020	1024	1027	1033	1038	1041
		1044	1047	1050	1053	1056	1059	1062	1065
		1068	1072	1075	1078	1081	1084	1087	1090
		1093	1096	1099	1103	1107	1110	1113	1116
		1119	1122	1125	1130	1135	1139	1142	1145
		1149	1152	1156	1160	1163	1167	1170	1173
		1176	1180	1184	1190	1197	1204		
clc$system_logging_active_name	4910	4978							
cle$name_not_a_keyword_value	39	5332/P							
clv$command_statistics_enabled	4808	4994							
clv$secure_logging_activated	4881	4977							
clv$system_logging_activated	4914	4978							
deadstart_only	4791	5303							
default	4296	5270/M							
dfc$client_job_heap_size	1849	1843							
dfc$client_job_list_size	1929	1794	1795	1932					
dfc$command_record_bytes	3461	3469							
dfc$division_overwrite_words	3448	3476							
dfc$esm_command_record_size	3469	3477							
dfc$esm_header_record_size	3470	3477							
dfc$esm_maintenance_buf_size	3449	3480							
dfc$esm_memory_base_shift	3455	3477	3478	3478					
dfc$header_record_bytes	3460	3470							
dfc$max_esm_memory_size	3450	3479							
dfc$max_family_parameters	1218	1214							
dfc$max_family_ptr_array_size	1248	1251							
dfc$max_job_list_p_array_size	1935	1938	5140						
dfc$max_number_of_mainframes	3457	3442							
dfc$maximum_family_lists	1227	1218	5142						
dfc$maximum_lifetime	1870	1867							
dfc$min_data_record_bytes	3465	3476							
dfc$min_esm_division_size	3475	3479							
dfc$served_family_list_size	1242	1219	1245						
dft$client_job_list_entry	1799	1796							
dft$client_job_list_index	1932	1925							
dft$client_job_list_pointer	1793	1791							
dft$client_job_list_root	1784	1775							
dft$client_job_space	1817	1814							
dft$client_mainframe_file	1758	1763							
dft$family_access_kinds	1861	1824	1863						
dft$family_list	1231	1230							
dft$family_pointer_index	1251	1237							
dft$job_heap	1841	1837							
dft$job_list_pointer_array	1790	1786							
dft$job_list_ptr_array_index	1938	1924							
dft$lifetime	1867	1803	3126						
dft$mainframe_file_header	1765	1759							
dft$mainframe_set	3442	3392	3393	3513	3514				
dft$served_family_list_index	1245	1238							

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dft$server_lifetime	3126	1771						
dft$server_state	3131	1770	3134					
dfv$file_server_debug_enabled	1733	5034						
dfv$file_server_info_enabled	1736	5036						
dfv$job_recovery_enabled	1739	5035						
dfv$maximum_client_job_lists	1744	5141						
dfv$number_served_family_lists	3208	5143						
dmp$fetch_debug_option_value	1619	5232						
dmp$store_debug_option_value	1674	5338						
dmt$global_file_name	2947	2600	2720					
dmt$stored_fmd	2956	2848						
dmt$system_file_id	2016	1994	2465	3424				
dmv$maximum_allocation_size	4927	5150						
dmv$permanent_file_overflow	3213	5091						
dmv$pf_sparse	4866	5149						
dmv$quick_deadstart	4928	5093						
dmv$recycle_device_log	3216	5098						
dmv$space_messages_to_console	4809	5152						
dmv$temporary_file_overflow	3219	5112						
dmv$test_recovery	3222	5113						
dmv$trim_files	4810	5046						
dmv$volume_class_kludge	4811	5120						
dpv$display_delay	4812	5020						
dpv$enable_console_bell	3225	5154						
dsv$display_deadstart_messages	4813	5021						
dsv$ignore_image	4815	5044						
dsv$unload_deadstart_tape	3229	5047						
error_string	5356	5375/M	5376/M					
fmt$file_label	2961	2876	2963					
fsc$reserved_damage_symptom_9	2056	2083						
fst$cycle_damage_symptom	2052	2037						
fst$cycle_damage_symptoms	2037	1985	2333	2718				
gft$file_descriptor_index	2031	2021						
gft$system_file_identifier	2020	2016	4018					
gft$table_residence	2034	2022						
halfword_p	4799	5200	5220	5317/M				
i	5357	5363	5364	5364	5365/M	5365		
iiv$condition_handler_trace	4816	5043						
iiv$output_option	4818	5089						
index	5183	5193	5193	5194/S	5195/S	5198/S	5200/S	5202/S	5204/S
		5207	5208/M	5210/M	5210				
integer_p	4797	5198	5218	5315/M					
iop$fetch_debug_option_value	1681	5237						
iop$store_debug_option_value	1688	5343						
iot$io_error	4082	3425	4048					
iov$unused_boolean	4820	5125						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmc$detached_job_wait_time_max	3295	3292						
jmc$highest_det_job_wait_time	3305	3295	3306					
jmc$highest_service_accumulator	3896	3897						
jmc$highest_working_set_size	4114	4105	4115	4117	4119	4121		
jmc$ies_job_swapped	3681	3690						
jmc$ies_swapin_in_progress	3680	3689						
jmc$iss_idle_tasks_initiated	3696	3723						
jmc$iss_swapin_io_complete	3721	3724						
jmc$iss_swapin_requested	3717	3724						
jmc$iss_swapout_complete	3716	3723						
jmc$iss_swapped_io_cannot_init	3707	3734						
jmc$iss_swapped_no_io	3698	3733						
jmc$keyword_offset_maximum	3322	4106						
jmc$kjl_maximum_entries	1274	1267	1268	3848				
jmc$kol_maximum_entries	1284	1269						
jmc$max_active_jobs	1265	5124						
jmc$max_ajl_ord	1266	1265	3489					
jmc$max_dispatching_control	3640	3644						
jmc$max_dispatching_priority	3562	3522	3525	3526				
jmc$maximum_job_classes	3831	3834	5126					
jmc$maximum_job_count	1281	1274	5059					
jmc$maximum_output_count	1291	1284	5060					
jmc$maximum_service_classes	3914	3917	5128					
jmc$min_dispatching_control	3639	3643						
jmc$minimum_job_classes	3830	5126						
jmc$minimum_service_classes	3913	5128						
jmc$null_service_class	3907	3908						
jmc$priority_p1	3576	3523						
jmc$priority_p10	3585	3524						
jmc$priority_p14	3589	3524						
jmc$priority_p8	3583	3523						
jmc$required_offset	3320	4120						
jmc$reserved_ajls	1270	1265						
jmc$service_accumulator_maximum	3888	3885						
jmc$system_default_offset	3321	3322	4122					
jmc$system_supplied_name_size	1883	1880						
jmc$unlimited_offset	3318	3296	3307	3898	4116			
jmc$unspecified_offset	3319	4118						
jmc$working_set_size_maximum	4105	4102						
jmt$ajl_ordinal	3489	3361	4180					
jmt$delayed_swapin_work	3506	3391	3510					
jmt$detached_job_wait_time	3292	3277						
jmt$dispatching_control_index	3643	3600	3610					
jmt$dispatching_controls	3613	3611						
jmt$dispatching_priority	3522	3373	3601	3602	3603	3615		
jmt$ijl_block_index	3341	3337						
jmt$ijl_block_number	3340	3336						
jmt$ijl_dispatching_control	3599	3374						
jmt$ijl_entry_status	3676	3360						
jmt$ijl_ordinal	3335	3267	3380	3408	3950	3951	4011	4040	4185

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES			
jmt$ijl_page_fault_count	3750	3745	3746	3747	
jmt$ijl_page_stats	3744	3740			
jmt$ijl_service_class_stats	3738	3395			
jmt$ijl_statistics	3783	3394			
jmt$ijl_swap_count	3759	3755	3756		
jmt$ijl_swap_counts	3754	3414	3741		
jmt$ijl_swap_status	3694	3363	3364	3365	
jmt$initiated_job_list_entry	3357	3266	3975		
jmt$input_file_location	3868	3863			
jmt$job_abort_disposition	3877	3861			
jmt$job_class	3834	3419			
jmt$job_control_block	3248	3234			
jmt$job_mode	1873	1802	3376		
jmt$job_priority	3837	3416	3417		
jmt$job_recovery_disposition	3880	3862			
jmt$job_system_id	4098	3263			
jmt$kjl_index	3848	3362	4098		
jmt$queue_file_ijl_information	3860	3401			
jmt$scheduling_data	3407	3385			
jmt$service_accumulator	3885	3409	3410	3411	
jmt$service_class_index	3917	3420			
jmt$swap_data	3423	3387			
jmt$swapout_reasons	3920	3415			
jmt$swapped_job_entry	3935	3286	3432	3976	
jmt$system_supplied_name	1880	1800	3261	3358	
jmt$task_time_slice	3653	3633	3634		
jmt$time_slice_values	3632	3617			
jmt$user_supplied_name	1892	1801	3262		
jmt$working_set_size	4102	3273	3274		
jmv$enable_queue_file_access	4821	5028			
jmv$max_think_time	4929	5072			
jmv$maximum_job_classes	4140	5127			
jmv$maximum_known_jobs	4144	5059			
jmv$maximum_known_outputs	4153	5060			
jmv$maximum_service_classes	4159	5129			
jmv$min_think_time	4930	5081			
jmv$scan_idle_dispatch_interval	4163	4991			
jmv$sched_memory_wait_factor	4166	5101			
jmv$scheduler_wait_time	4170	5100			
jmv$swap_file_allocation_size	4822	5105			
jmv$swap_jobs_in_long_wait	4175	5106			
jsc$isqi_swapped_io_completed	3955	3957			
jsc$isqi_swapped_io_not_init	3954	3957			
jst$changed_asid_entry	3998	3989			
jst$ijl_swap_queue_id	3954	3949			
jst$ijl_swap_queue_link	3948	3369			
jst$io_control_information	3962	3388			
jst$swap_file_descriptor	3974	3389			
jst$swapped_page_descriptor	3983	3981			
jst$swapped_page_descriptors	3980	3977			
jsv$enable_debug_code	4191	5024			

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jsv$enable_swap_file_statistics	4195	5030							
jsv$enable_swap_resident	4931	5031							
jsv$enable_swap_resident_no_io	4932	5032							
jsv$free_working_set_on_swapout	4202	5038							
jsv$halt_on_swapin_failure	4199	5042							
jsv$max_pages_first_swap_task	4209	5063							
jsv$max_time_swap_io_complete	4212	5069							
jsv$max_time_swap_io_not_init	4215	5074							
jsv$maximum_pages_to_swap	4206	5062							
jsv$think_expiration_time	4219	5114							
jsv$write_stale_pages	4933	5123							
lname	5358	5362/M	5363	5364	5364	5365/M	5365	5370	5370
		5376	5377/P						
lov$enable_source_type_checking	4823	5029							
lov$ignore_param_verification	4824	5045							
lp	5369	5369	5373	5379					
ma	4311	5270/M							
ma_tt	4289	5270/S							
max	4794	5309							
max_name_length	4957	4792	5358						
min	4793	5309							
minimum	1331	5191	5312	5327/M					
mlv$wire_mli_tables	4934	5121							
mmc$pmo_r1_stack	1299	5273/S	5282/S	5284/S					
mmc$pmo_r2_stack	1299	5276/S							
mmc$pmo_r3_stack	1299	5279/S							
mmc$pq_avail	1489	1535							
mmc$pq_free	1488	1547							
mmc$pq_job_fixed	1529	1536	1548						
mmc$pq_job_working_set	1531	1548	1549						
mmc$pq_shared_first_site	1539	1543							
mmc$pq_shared_num_sites	1540	1543							
mmc$pq_shared_other	1498	1538							
mmc$pq_shared_site_01	1500	1539							
mmc$pq_shared_site_25	1524	1544							
mmc$pq_shared_task_service	1493	1537	5191/S	5312/S	5327/S				
mmc$pq_swapped_io_error	1527	1547							
mmc$pq_wired	1491	1534							
mmc$ring_crossing_offset	1313	5274	5277	5280	5283	5285			
mmt$active_segment_table_entry	4008	3986	4024	4047					
mmt$ast_index	4030	3431	4001						
mmt$global_page_queue_index	1547	1338							
mmt$global_page_queue_list	1338	4265	4293						
mmt$global_page_queue_list_ent	1328	1338							
mmt$job_page_queue_index	1548	1339	3937						
mmt$job_page_queue_list	1339	3386							
mmt$link	1448	1325	4009	4037	4038				
mmt$locked_page	4059	4043							
mmt$manage_memory_utility	4309	4281							

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmt$memory_reserve_request	4088	3379						
mmt$mmu_gpql_default	4293	4310						
mmt$mmu_ma_data	4307	4311						
mmt$mmu_ma_info	4295	4307						
mmt$mmu_memory_attributes	4288	4291	4292	4307				
mmt$mmu_value_types	4290	4297						
mmt$page_age	4066	4046	4070	4070				
mmt$page_frame_index	1454	1450	1450	3963	3965	3966	3967	4090	4091
mmt$page_frame_queue_id	1549	3964	4017	4041				
mmt$page_frame_table_entry	4036	3984	4052					
mmt$page_map_offsets	1301	4923	4923					
mmt$page_map_offsets_ord	1298	1301						
mmt$page_queue_list_entry	1324	1329	1339					
mmu_boolean	4290	4302						
mmu_byte	4290	4300						
mmu_integer	4290	4298						
mmv$advise_in_aio_limit	4935	4980						
mmv$age_interval_ceiling	4936	4981						
mmv$age_interval_floor	4937	4982						
mmv$aggressive_aging_level	4227	4983						
mmv$aggressive_aging_level_2	4228	4984						
mmv$aging_algorithm	4235	4985						
mmv$aio_limit	4238	4986						
mmv$assign_multiple_pages	4246	4975						
mmv$check_queues	4938	5082						
mmv$disable_write_for_perf_meas	4251	5019						
mmv$file_allocation_interval	4254	5033						
mmv$gpql	4265	5191	5312	5327/M				
mmv$jws_queue_age_interval	4273	5049						
mmv$manage_memory_utility	4281	5270/M						
mmv$max_pages_no_file	4322	5061						
mmv$max_segment_length	4836	5068						
mmv$maximum_180_memory	4830	5055						
mmv$maxws_aio_threshold	4939	5073						
mmv$min_avail_pages	4325	5075						
mmv$multi_page_write	4332	5084						
mmv$no_memory_buffering	4338	5087						
mmv$page_map_offsets	4923	5273	5276	5279	5282	5284		
mmv$periodic_call_interval	4344	5083						
mmv$read_tu_execute	4347	5094						
mmv$read_tu_read_write	4346	5095						
mmv$shadow_by_segnum	4831	5135						
mmv$shared_pages_in_jws	4351	5102						
mmv$shared_queue_age_interval	4358	5103						
mmv$swapping_aic	4940	5104						
mmv$tick_time	4362	5264/M	5270					
mmv$write_aged_out_pages	4370	5122						
mtv$aborted_task_threshold	4377	4979						
mtv$automatic_unstep_resume	4381	4988						
mtv$halt_cpu_ring_number	4389	5039						
mtv$halt_on_proc_malf	4374	5041						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mtv$mx_ajl_entries	4837	5124						
mtv$mx_segments	4838	5056						
mtv$mx_tasks	4839	5071						
mtv$processor_due_threshold	4392	5092						
mtv$scb_vector_sim_attribute	4840	5118						
mtv$sys_core_init_complete	4395	5303						
mtv$system_haltring	4388	5110						
mtv$xp_initial_value	4400	5272/M	5275/M	5278/M	5281			
name	4792	5194	5370	5370				
name	5182	5192	5194/M	5195	5214/P	5232/P	5237/P	
name	5291	5300/P	5332/P	5338/P	5343/P			
name	5351	5362						
nav$disable_network_relays	4863	5133						
normal	1627	5176/M	5215	5233	5246/M	5301	5339	
offset	1424	5272/M	5275/M	5278/M				
osc$aging_interval_maximum	4126	4129						
osc$call_instruction	4582	4590						
osc$data_read	4581	4590						
osc$free_running_clock_maximum	1435	1432						
osc$invalid_ring	1370	1410						
osc$max_name_size	1254	1258	1261	2391	2991			
osc$max_page_frames	1458	1326	1332	1454	3426	3427	3936	3938	4010
osc$max_page_size	2265	2261						
osc$max_page_table_entries	1459	1462						
osc$max_processor_model_index	4607	4622						
osc$max_ring	1369	1410	1411					
osc$max_segment_length	1393	1416						
osc$max_status_condition_code	1642	1638	1654					
osc$max_string_size	1658	1661	1664	1669	1697	1706		
osc$max_tasks	1565	1562	5071					
osc$maximum_offset	1392	1393	1413	1413	1414			
osc$maximum_segment	1391	1412						
osc$min_page_size	2264	2261						
osc$min_ring	1368	1411						
osc$status_parameter_delimiter	1646	5333/P						
osc$task_time_slice_maximum	3664	3667						
osp$append_status_parameter	1695	5333						
osp$set_status_abnormal	1703	5304	5310	5332	5377			
ost$aging_interval	4129	3275	3276					
ost$asid	1356	1352	3370	3988	3999	4000	4015	
ost$binary_unique_name	2173	2163	2423	2947				
ost$cp_time	3771	3739	3784					
ost$cp_time_value	3769	3270	3271	3412	3772	3773		
ost$date_time	2966	2676	2716	2726	2805	2808	2810	2997	3004
		3029	3030	3031				
ost$debug_code	4581	4569						
ost$debug_list	4577	4460						
ost$debug_list_entry	4568	4577						
ost$debug_mask	4587	4459						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$exchange_package	4409	4400							
ost$family_name	1948	1820	1943	2352	2356	2361	2367	2372	2378
ost$flags	4466	4416							
ost$frame_descriptor	4545	4560							
ost$free_running_clock	1432	3269	3278	3381	3382	3383	3384	3418	3428
		3429	3430	3604	3616	4014			
ost$global_task_id	1556	3265	3375	3404					
ost$halfword	1438	4144	4153						
ost$heap	1919	1760	1836	2531					
ost$key_lock_value	1405	1402	4483	4485					
ost$keypoint_class	4498	4429	4500						
ost$keypoint_mask	4500	4432							
ost$minimum_save_area	4555	4421	4530						
ost$monitor_condition	4504	4511							
ost$monitor_conditions	4511	4422	4426	4535					
ost$name	1261	1215	1231	1767	1854	1858	1892	1946	1948
		2289	2422	2431	2985	2990			
ost$p_register	4481	4410	4556						
ost$page_id	1464	1474							
ost$page_size	2261	2242	4597						
ost$page_table_entry	1469	1478	3985						
ost$page_table_index	1462	1478	4044						
ost$paging_statistics	3807	3785							
ost$processor_model_definition	4610	4603	4625						
ost$processor_model_index	4622	4624							
ost$processor_model_number	2191	2175	3102	4611					
ost$processor_serial_number	2269	2174	3103						
ost$pva	1421	4454	4472	4486					
ost$register_number	4477	4451	4541	4549	4550	4551			
ost$ring	1410	1422	4471						
ost$segment	1412	1423	1773	4449	4571				
ost$segment_offset	1413	1353	1424	4572	4574				
ost$signature_lock	2090	1774	1830						
ost$stack_frame_save_area	4529	4563							
ost$status	1626	1621	1676	1683	1690	1698	1707	1717	5174
		5185	5189	5244	5253	5294	5298	5353	
ost$status_condition_code	1654	1629	1650	1705	3009				
ost$status_identifier	1712	1704							
ost$string	1667	1630							
ost$string_size	1661	1668							
ost$system_virtual_address	1351	4049							
ost$task_index	1562	1557	4077	4078					
ost$task_time_slice	3667	3653							
ost$top_of_stack_pointer	4469	4461							
ost$trap_enable	4524	4418							
ost$user_condition	4514	4521							
ost$user_conditions	4521	4420	4424	4533	4562				
ost$user_identification	1941	2675	3264						
ost$user_name	1946	1821	1942	2368	2374	2381			
ost$valid_ring	1411	4461							
ost$virtual_machine_identifier	4491	4412	4414	4557					

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$x_register	4478	4451	4541						
osv$catalog_name_security	4841	4989							
osv$debug	4941	4997	4998	4999	5000	5001	5002	5003	5004
		5005	5006	5007	5008	5009	5010	5011	5012
osv$default_pit	4842	5015							
osv$default_sit_value	4942	5016							
osv$delete_unreconciled_files	4843	5017							
osv$disk_fault_simulation	4943	5131							
osv$dump_when_debug	4844	5022							
osv$emergency_intervention	4845	5132							
osv$enable_hyperchannel	4846	5027							
osv$global_processor_model_info	4603	5263	5264						
osv$keypoint_enable	4945	5050							
osv$page_size	4597	5273	5276	5279	5282	5284			
osv$reconcile_permanent_files	4847	5096							
osv$recover_at_all_costs	4848	5097							
osv$reorganize_permanent_files	4849	5099							
osv$special_aam_trap	4695	5144							
osv$trap_task_errors	4946	5145							
osv$validate_active_sets	4850	5117	5148						
osv$validate_permanent_files	4851	5116							
p	5256	5281/M	5282/M	5284/M					
pfc$account	2344	2354							
pfc$attached_pf_awaiting_client	2415	1993							
pfc$attached_pf_entry_unused	1972	1967							
pfc$attached_pf_entry_valid	1973	1965							
pfc$attached_pf_in_job_recovery	2415	1999							
pfc$attached_pf_normal	2414	1993							
pfc$catalog_object	2584	2576							
pfc$charge_change	2316	2330							
pfc$connected_file_device	3087	2747							
pfc$cycle_expansion_count	2689	2690							
pfc$cycle_number_change	2315	2324							
pfc$delete_damage_change	2316	2332							
pfc$execute	2394	2396	2399						
pfc$external_catalog	3039	2598							
pfc$family	2344	2350							
pfc$family_name_index	2282	2283							
pfc$family_path_index	2136	2137							
pfc$file_object	2583	2570							
pfc$free_cycle_entry	2732	2711							
pfc$free_log_entry	2682	2672							
pfc$free_mainframe_entry	2905	2896							
pfc$free_object	2583	2568							
pfc$free_permit_entry	2642	2482	2632						
pfc$highest_cycle	2295	2301							
pfc$internal_catalog	3039	2596							
pfc$interstate_link_device	3088	2749							
pfc$local_queue_device	3089	2751							
pfc$log_change	2315	2328							

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
pfc$log_device	3090	2753						
pfc$log_expansion_count	2649	2650						
pfc$lowest_cycle	2295	2299						
pfc$magnetic_tape_device	3091	2755						
pfc$mass_storage_device	3092	2757						
pfc$master_catalog_name_index	2283	2284						
pfc$master_catalog_path_index	2137	2138						
pfc$max_amd_size	2821	2824						
pfc$maximum_archive_count	2784	2788	2789					
pfc$maximum_catalog_depth	2140	2143	2144					
pfc$maximum_cycle_count	3051	2694	3054					
pfc$maximum_cycle_number	2277	2293						
pfc$maximum_file_label_size	2867	2870						
pfc$maximum_fmd_size	2843	2846						
pfc$maximum_log_count	2651	2655	2656					
pfc$maximum_object_count	2539	2544	2545					
pfc$maximum_permit_count	2609	2615	2616					
pfc$maximum_retention	2280	2310						
pfc$member	2345	2376						
pfc$memory_resident_device	3093	2761						
pfc$minimum_cycle_number	2276	2293						
pfc$minimum_retention	2279	2310						
pfc$network_device	3094	2763						
pfc$normal_cycle_entry	2732	2713						
pfc$normal_log_entry	2682	2674						
pfc$normal_mainframe_entry	2906	2898						
pfc$normal_permit_entry	2642	2484	2634					
pfc$null_device	3095	2765						
pfc$password_change	2314	2322						
pfc$permit_expansion_count	2610	2611						
pfc$pf_name_change	2314	2320						
pfc$pipeline_device	3096	2767						
pfc$project	2345	2359						
pfc$public	2344	2348						
pfc$purged_catalog_object	2585	2576						
pfc$purged_cycle_entry	2733	2713						
pfc$purged_file_object	2584	2570						
pfc$read	2393	2396	2399					
pfc$retention_change	2315	2326						
pfc$rhfam_device	3097	2769						
pfc$set_path_index	2135	2136						
pfc$specific_cycle	2296	2303						
pfc$terminal_device	3098	2771						
pfc$user	2345	2365						
pfc$user_account	2345	2370						
pft$access_count	3034	2677	3032					
pft$amd	3013	2830	3015					
pft$amd_locator	2833	2812						
pft$amd_size	2824	2834						
pft$application_info	2391	2488	2638					
pft$archive_count	2788	2799						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES						
pft$archive_entry	2803	2795						
pft$archive_entry_version	2978	2804						
pft$archive_identification	2984	2806						
pft$archive_list	2787	2790						
pft$archive_list_locator	2798	2722						
pft$archive_media_identifier	2989	2986						
pft$attach_status	2735	2717						
pft$attached_pf_entry	1977	1975						
pft$attached_pf_entry_type	1972	1964						
pft$attached_pf_header	1963	1961						
pft$attached_pf_recovery_state	2414	1992						
pft$attached_pf_table	1961	1960						
pft$catalog_file	2510	2515	2823					
pft$catalog_header	2524	2519						
pft$catalog_heap	2531	2512	2532					
pft$catalog_object_locator	2594	2577						
pft$catalog_types	3039	2595						
pft$change_descriptor	2318	2317						
pft$change_type	2314	2319						
pft$charge_id	3019	2565						
pft$checksum	3044	2518	2550	2621	2661	2699	2794	2829	2853
		2875	2890	3047				
pft$complete_path	2146	2147						
pft$cycle_count	3054	2704						
pft$cycle_device_information	2741	2727						
pft$cycle_entry	2708	2700						
pft$cycle_entry_types	2732	2710						
pft$cycle_list	2693	2695						
pft$cycle_list_locator	2703	2574						
pft$cycle_number	2293	1987	2304	2325	2678	2714		
pft$cycle_options	2295	2298						
pft$cycle_statistics	3028	2715						
pft$data_residence	3077	2723						
pft$device_class	3083	2746						
pft$file_label_locator	2879	2721						
pft$file_label_size	2870	2880						
pft$fmd	2848	2849	2854					
pft$fmd_locator	2857	2599	2719					
pft$fmd_size	2846	2858						
pft$group	2346	2635						
pft$group_types	2344	2347	2485					
pft$internal_catalog_name	2164	2165	2463	2464	3111	3112		
pft$internal_cycle_path	2152	1988	2157					
pft$internal_name	2163	2149	2153	2164	2563	2709		
pft$internal_path	2149	2150	2154					
pft$log	2312	2329	2572					
pft$log_count	2656	2666						
pft$log_entry	2670	2662						
pft$log_entry_types	2682	2671						
pft$log_list	2654	2657						
pft$log_list_locator	2665	2573						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pft$mainframe_count	2913	2909							
pft$mainframe_entry_type	2905	2895							
pft$mainframe_list_locator	2908	2725							
pft$mainframe_usage_entry	2894	2724	2891						
pft$name	2289	2146	2291	2308	2321	2462	2526	2562	3110
pft$object_count	2545	2555	2557	2558					
pft$object_entry	2561	2551							
pft$object_list	2543	2546							
pft$object_list_locator	2554	2527	2597						
pft$object_types	2583	2567	2587						
pft$p_attached_pf_entry	1975	1966							
pft$p_attached_pf_table	1960	1833							
pft$p_complete_path	2147	1986							
pft$p_queued_catalog	2492	1832	2457	2458					
pft$p_queued_catalog_table	2494	1831							
pft$p_queued_internal_catalog	3107	2468	3116						
pft$password	2308	2323	2571						
pft$path	2291	2159							
pft$permit_count	2616	2626							
pft$permit_entry	2630	2622	3115						
pft$permit_entry_types	2642	2481	2631						
pft$permit_list	2614	2617							
pft$permit_list_locator	2625	2564							
pft$permit_options	2393	2340							
pft$permit_selections	2340	2486	2636						
pft$physical_amd	2828	2826							
pft$physical_archive	2793	2787	2791						
pft$physical_catalog_header	2517	2511	2522						
pft$physical_cycle	2698	2693	2696						
pft$physical_file_label	2874	2872							
pft$physical_fmd	2852	2850							
pft$physical_log	2660	2654	2658						
pft$physical_mainframe_usage	2889	2916							
pft$physical_object	2549	2543	2547						
pft$physical_permit	2620	2614	2618						
pft$queued_catalog	2456	2492	2494						
pft$queued_internal_catalog	3109	3107							
pft$queued_permit_entry	2480	2470							
pft$relative_cell_pointer	2923	2556	2627	2667	2705	2800	2835	2859	2881
		2910							
pft$release_candidate	2994	2811							
pft$retention	2310	2327							
pft$retrieval_status	3003	2809							
pft$sequence	2925	2927	2930						
pft$sequence_record	2929	2933							
pft$sfid_status	1991	1978							
pft$share_options	2399	2341	2400	2738					
pft$share_requirements	2341	2487	2637						
pft$share_selections	2400	1982							
pft$usage_count	2777	2736	2737	2738	2900	2901			
pft$usage_options	2396	2397	2737						

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pft$usage_selections	2397	1981							
pfv$binary_catalog_search	4854	5155							
pmc$mainframe_id_size	3143	3140							
pmc$processor_model_number_size	3151	3143	3148						
pmc$processor_model_type_size	4632	4629							
pmc$processor_serial_num_size	3201	3144	3198						
pmt$binary_mainframe_id	3101	1768	2899	3268					
pmt$cpu_model_number	2251	2240	2247						
pmt$cpu_serial_number	2254	2241	2246						
pmt$mainframe_id	3140	1769							
pmt$processor_model_number	3148	4612							
pmt$processor_model_type	4629	4613	4614						
pmt$sense_switches	4135	3279							
pmt$vector_capability	4686	4619							
pmv$constrain_meape_segments	4856	5137							
pmv$debug_logging_enabled	4855	5136							
pmv$quantum	4698	5263/M							
pva	4472	5272/M	5275/M	5278/M					
quantum	4615	5263							
rav$deadstart_intervention	4857	4996							
rav$development_deadstart	4858	5018							
rav$network_activation	4859	5085							
rav$system_activation	4860	5138							
search_value_table	5350	5214	5300	5382					
sft$counter	3817	3280	3282	3283	3285	3786	3787		
sfv$dynamic_file_space_limits	4947	5023							
status	5174	5176/M							
status	5185	5214/P	5215						
status	5244	5246/M							
status	5294	5300/P	5301	5305/P	5310/P	5332/P	5333/P		
status	5353	5377/P							
stt$set_name	2422	2461							
syc$min_ecc	1570	1571	1572	1573	1574	1575			
syc$min_ecc_commands	1572	1580	1583	1586	1589	1592	1595	1598	1601
		1604	1607	1610					
sye$not_changeable_after_ds	1607	5304/P							
sye$range_error	1592	5310/P							
sye$unknown_parameter_name	1601	5377/P							
syp$fetch_system_const_from_ssr	5173	5177							
syp$fetch_system_constant	5181	5239							
syp$save_system_const_in_ssr	5243	5247							
syp$set_processor_attributes	5251	5286							
syp$store_system_constant	5290	5346							
syt$debug_output_disposition	4728	4721							
syt$dflt_debug_output_disposal	4716	4711							
syv$allow_jr_test	4948	4987							
syv$clone_enabled	4867	4993							
syv$debug_job_recovery	4949	5013							

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
syv$debugger_page_wait_lines	4707	5151							
syv$dflt_debug_output_disposal	4711	5153							
syv$enable_fault_injection	4737	5025							
syv$enable_heap_trace	4701	5026							
syv$halt_on_exit_with_io	4861	5156							
syv$job_recovery_option	4739	5048							
syv$mandatory_dualstate	4754	5054							
syv$max_debug_output_lines	4758	5130							
syv$nosve_internal_operations	4862	5086							
syv$system_job_multiprocessing	4950	5111							
syv$user_templates	4864	5139							
syv$verify_heap_linkage	4762	5119							
tab_p	5188	5214/P	5216	5218	5220	5222	5224		
tab_p	5297	5300/P	5303	5309	5309	5313	5315/M	5317/M	5319/M
		5322/M	5324/M						
tab_p	5352	5381/M							
table_i	5359	5361/M	5370/S	5370/S	5371/M	5371	5372	5381/S	
temp_status	5189	5232/P	5233	5237/P					
temp_status	5298	5338/P	5339	5343/P					
temp_workaround_min_sws	4970	5077	5191/M	5312/M	5327				
tick_time	4616	5264							
tmt$task_queue_link	4076	4045							
tmv$cycle_delay_time	4766	4995							
tmv$dedicate_a_cpu_to_nos	4770	5014							
tmv$display_actual_priority	4865	5146							
tmv$halt_on_hung_task	4774	5040							
tmv$long_wait_force_swap_time	4951	5053							
tmv$long_wait_swap_time	4952	5052							
tmv$system_debug_ring	4775	5107							
tmv$system_debug_segment	4776	5108							
tmv$system_error_hang_count	4777	5109							
tmv$timed_wait_not_queued	4784	5147							
tos_registers	4461	5272/M	5275/M	5278/M					
v_boolean	4795	4802	4977	4978	4987	4988	4989	4993	4994
		4996	5013	5014	5017	5018	5019	5021	5022
		5023	5024	5025	5026	5027	5028	5029	5030
		5031	5032	5034	5035	5036	5037	5038	5040
		5041	5042	5043	5044	5046	5047	5051	5054
		5084	5085	5086	5087	5088	5090	5091	5093
		5096	5097	5098	5099	5102	5106	5111	5112
		5113	5115	5116	5117	5119	5120	5121	5123
		5125	5131	5132	5133	5134	5135	5136	5137
		5138	5139	5144	5145	5146	5148	5149	5152
		5154	5155	5156	5157	5158	5159	5160	5161
		5162	5163	5164	5165	5166	5167	5203	5223
		5320							
v_byte	4795	4800	4976	4981	4982	5039	5110	5118	5153
		5201	5221	5318					
v_halfword	4795	4798	5056	5059	5060	5071	5076	5094	5095

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

System Constant Management
search_value_table

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		5105	5109	5124	5126	5128	5199	5219	5316
v_integer	4795	4796	4975	4979	4980	4983	4984	4985	4986
		4990	4995	4997	4998	4999	5000	5001	5002
		5003	5004	5005	5006	5007	5008	5009	5010
		5011	5012	5015	5016	5020	5033	5045	5048
		5049	5050	5052	5053	5055	5057	5058	5061
		5062	5063	5068	5069	5070	5072	5073	5074
		5075	5081	5082	5083	5089	5092	5100	5101
		5103	5104	5107	5108	5114	5122	5130	5140
		5142	5147	5150	5151	5197	5217	5314	
value	5184	5198/M	5200/M	5202/M	5204/M	5218/M	5220/M	5222/M	5224/M
		5232/P	5237/P						
value	5293	5309	5309	5315	5317	5319	5321	5338/P	5343/P
value_table	4974	5194	5196	5198	5200	5202	5204	5370	5370
		5381							
value_table_length	4958	4974	5193	5207	5372				
value_table_type	4790	4974	5188	5297	5352				
value_type	4795	5196	5216	5313					

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

```
    3 MODULE tmm$dispatcher;
    4
    5
    6 {
    7 {  PURPOSE:
    8 {     This module is used for managing the creation and deletion of tasks,
    9 {     the creation and deletion of jobs, the status of tasks, and the
   10 {     selection of tasks to execute.
   11 {
   12
   13   ?VAR
   14      debug: boolean := FALSE?;



o  5233
o  5234   PROCEDURE [INLINE] gfp$mtr_get_fde_p (sfid: gft$system_file_identifier;
o  5235         ijle_p: ^jmt$initiated_job_list_entry;
o  5236     VAR fde_p: gft$file_desc_entry_p);
o  5237
o  5238
o  5285
o  5286   PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
o  5287         ijle_p: ^jmt$initiated_job_list_entry;
o  5288     VAR fde_p: gft$locked_file_desc_entry_p);
o  5289
o  5290
o  5351
o  5352 { PURPOSE:
o  5353 {   This is the monitor mode procedure to change the entry status of a job.  The caller
o  5354 {   of procedure must set the PTL lock if the entry status change is a SWAPPED/NOT SWAPPED
o  5355 {   transition because the swapped job counts will be changed.
o  5356
o  5357   PROCEDURE [INLINE] jmp$change_ijl_entry_status
o  5358     (     ijle_p: ^jmt$initiated_job_list_entry;
o  5359           new_entry_status: jmt$ijl_entry_status);
o  5360
o  5361     VAR
o  5362       old_entry_status: jmt$ijl_entry_status;
o  5363
o  5364     old_entry_status := ijle_p^.entry_status;
o  5365
o  5366     jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] :=
o  5367           jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] + 1;
o  5368
o  5369     ijle_p^.entry_status := new_entry_status;
o  5370
o  5371     IF (old_entry_status <= jmc$ies_swapin_in_progress) AND
o  5372           (new_entry_status > jmc$ies_swapin_in_progress) THEN
o  5373       jmp$increment_swapped_job_count (ijle_p);
o  5374
o  5375     ELSEIF (old_entry_status > jmc$ies_swapin_in_progress) AND
o  5376           (new_entry_status <= jmc$ies_swapin_in_progress) THEN
o  5377       jmp$decrement_swapped_job_count (ijle_p);
o  5378     IFEND;
o  5379
```



```
o  5380   PROCEND jmp$change_ijl_entry_status;
o  5381   PROCEDURE [XREF] jmp$decrement_swapped_job_count(ijle_p: ^jmt$initiated_job_list_entry);
o  5382
   5385   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
   5386     VAR ijle_p: ^jmt$initiated_job_list_entry);
   5387
o  5395
o  5396   PROCEDURE [XREF]  jmp$increment_swapped_job_count (ijle_p: ^jmt$initiated_job_list_entry);
o  5397
   5400   PROCEDURE [XREF] jmp$ready_task_in_swapped_job (ijl_ord: jmt$ijl_ordinal;
   5401         ijle_p: ^jmt$initiated_job_list_entry);
   5402
o  5405
o  5406   PROCEDURE [XREF] jmp$set_job_terminated
o  5407     (     ijl_ordinal: jmt$ijl_ordinal;
o  5408           ijle_p: ^jmt$initiated_job_list_entry);
o  5409
   5412
   5413   PROCEDURE [XREF] jmp$set_scheduler_event (event: jmt$job_scheduler_events);
   5414
o  5441
o  5442   PROCEDURE [XREF] jmp$set_swapout_candidate
o  5443     (     ajl_ordinal: jmt$ajl_ordinal;
o  5444           swapout_reason: jmt$swapout_reasons);
o  5445
   5448
   5449   PROCEDURE [XREF] jmp$subsystem_priority_change
   5450     (     ijle_p: ^jmt$initiated_job_list_entry);
   5451
o  5454
o  5455   PROCEDURE [XREF] jmp$swap_non_dispatchable_job
o  5456     (     ajl_ordinal: jmt$ajl_ordinal);
o  5457
   5460 PROCEDURE [XREF] jmp$assign_ajl_entry (asid: ost$asid,
   5461       ijl_o: jmt$ijl_ordinal;
   5462       caller: 0 .. 010(16);
   5463       must_assign: boolean;
   5464   VAR ajl_o: jmt$ajl_ordinal;
   5465   VAR status: syt$monitor_status);
   5466
o  5473 PROCEDURE [XREF] jmp$free_ajl_entry
o  5474   (     ijle_p: ^jmt$initiated_job_list_entry;
o  5475         caller: 0 .. 10(16));
o  5476
   5479
   5480 PROCEDURE [XREF] jmp$free_ajl_with_lock
   5481   (     ijle_p: ^jmt$initiated_job_list_entry;
   5482         caller: 0 .. 10(16));
   5483
o  5486
o  5487   PROCEDURE [INLINE] jmp$unlock_ajl_with_lock
o  5488     (     ijle_p: ^jmt$initiated_job_list_entry);
o  5489
o  5490     VAR
o  5491       ajlo: jmt$ajl_ordinal;
o  5492
```

```
o  5493      ajlo := ijle_p^.ajl_ordinal;
o  5494      IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
o  5495        jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
o  5496      ELSE
o  5497        jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
o  5498      IFEND;
o  5499
o  5500    PROCEND jmp$unlock_ajl_with_lock;
o  5501
   5505
   5506    PROCEDURE [XREF] jmp$update_service_class_stats
   5507      (     ijle_p: ^jmt$initiated_job_list_entry);
   5508
o  5511    PROCEDURE [XREF] jsp$idle_tasks_complete (ijl_ordinal: jmt$ijl_ordinal);
o  5512
   5515
   5516    PROCEDURE [XREF] jsp$long_wait_aging (ijle_p: ^jmt$initiated_job_list_entry);
   5517
o  5520
o  5521    PROCEDURE [XREF] jsp$relink_swap_queue (ijl_ordinal: jmt$ijl_ordinal;
o  5522          ijle_p: ^jmt$initiated_job_list_entry;
o  5523          new_queue: jst$ijl_swap_queue_id);
o  5524
   5527
   5528    PROCEDURE [XREF] mtp$interrupt_processor (port_mask: ost$cpu_memory_port_mask);
   5529
o  5532  {REFERENCED BY ASSEMBLY CODE ALSO
o  5533
o  5534    VAR
o  5535      mtv$mli_status: [XREF] record
o  5536        ready: boolean,
o  5537        wait_inhibit: boolean,
o  5538      recend;
o  5539
o  5540    PROCEDURE [XREF] mmp$create_task (parent_xcb_p: ^ost$execution_control_block;
o  5541          xcb_p: ^ost$execution_control_block;
o  5542          ijle_p: ^jmt$initiated_job_list_entry);
o  5543
   5546
   5547    PROCEDURE [XREF] mmp$exit_task (xcb_p: ^ost$execution_control_block);
   5548
o  5551
o  5552    PROCEDURE [XREF] mmp$create_job (new_job_ajl_ordinal: jmt$ajl_ordinal;
o  5553          xcb_segnum_relative_jobs_as: ost$segment;
o  5554          parent_xcb_p: ^ost$execution_control_block;
o  5555          xcb_p: ^ost$execution_control_block);
o  5556
   5560
   5561    PROCEDURE [XREF] mmp$exit_job (xcb_p: ^ost$execution_control_block);
   5562
o  5565
o  5566     FUNCTION [INLINE] mmp$get_sdtx_entry_p
o  5567       (     xcb_p: ^ost$execution_control_block;
o  5568             segnum: ost$segment): ^mmt$segment_descriptor_extended;
o  5569
o  5570       mmp$get_sdtx_entry_p := #address (1, #segment (xcb_p),
```



```
o  5571          #SIZE (mmt$segment_descriptor_extended) * segnum + xcb_p^.sdtx_offset);
o  5572
o  5573     FUNCEND;
o  5574
   5577
   5578
   5579    PROCEDURE [INLINE] i#disable_traps (VAR old_te: 0 .. 3);
o  5598
o  5599    PROCEDURE [INLINE] i#move (source: ^cell;
o  5600          dest: ^cell;
o  5601          length: 0 .. 7fffffff(16));
o  5602
o  5603      VAR
o  5604        str1: ^string (65535),
o  5605        str2: ^string (65535);
o  5606
o  5608      IF length <> 0 THEN
o  5609        str1 := source;
o  5610        str2 := dest;
o  5611        str2^ (1, length) := str1^ (1, length);
o  5612        #SPOIL (str2^);
o  5613      IFEND;
o  5615    PROCEND i#move;
o  5616
o  5617    PROCEDURE [INLINE] i#restore_traps (old_te: 0 .. 3);
   5633    PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
o  5644
o  5645    PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
o  5646          condition: osc$max_status_condition_number + 1 .. 0fffffffff(16);
o  5647      VAR status: syt$monitor_status);
o  5648
   5658
   5659   PROCEDURE [XREF] osp$update_job_keypoint_mask
   5660     (     ijle_p: ^jmt$initiated_job_list_entry;
   5661           ijl_ordinal: jmt$ijl_ordinal);
   5662
   5663 {
   5664 {      NOTE: Callers of this procedure require that 'tmv$ptl_lock'
   5665 {            be XREFed or XDCLed in the module.
   5666 {
   5667
   5668    PROCEDURE [INLINE] tmp$check_ptl_lock
   5669      (VAR ptl_lock_set: boolean);
   5670
   5671      ptl_lock_set := tmv$ptl_lock.locked;
   5672
   5673    PROCEND tmp$check_ptl_lock;
   5674
   5675
   5676    PROCEDURE [INLINE] tmp$get_xcb_access_status
   5677      (     ijle_p: ^jmt$initiated_job_list_entry;
   5678            ijl_ordinal: jmt$ijl_ordinal;
   5679       VAR inhibit_access: boolean);
   5680
o  5742
o  5743
```

NOS/VE Task Management : dispatcher

```
o  5744   PROCEDURE [XREF] tmp$set_system_flag (task_id [input] : ost$global_task_id;
o  5745         flag_id [input] : ost$system_flag;
o  5746     VAR status [output] : syt$monitor_status);
o  5747
   5750
   5751   PROCEDURE [XREF] tmp$set_monitor_flag (task_id: ost$global_task_id;
   5752         flag_id: syt$monitor_flag;
   5753     VAR status: syt$monitor_status);
o  5756
o  5757   PROCEDURE [XREF] tmp$send_signal (taskid: ost$global_task_id;
o  5758         signal: pmt$signal;
o  5759     VAR status: syt$monitor_status);
o  5760
   5763
   5764   PROCEDURE [XREF] tmp$update_system_task_list (xcb_p: ^ost$execution_control_block);
   5765



   5767 [Define global and external variables.
   5768 [Define SMU Communications Block (SCB).
   5769
   5770   VAR
   5771     mtv$scb: [XREF] mtt$smu_communications_block;
o  5916
o  5917   VAR
o  5918     tmv$null_global_task_id: [READ] ost$global_task_id := [0, 0];
o  5919
o  5920 [Monitor segment table.]
o  5921
o  5922   VAR
o  5923     mtv$monitor_segment_table: [XREF] record
o  5924       st: ALIGNED [0 MOD 8] array [0 .. 4095] of mmt$segment_descriptor,
o  5925     recend;
o  5926
   5929
   5930   VAR
   5931     osv$cpus_logically_on: [XREF] 0 .. osc$max_number_of_processors;
   5932
o  5935
o  5936   VAR
o  5937     osv$cpus_physically_configured: [XREF] 1 .. osc$max_number_of_processors;
o  5938
   5941  VAR
   5942     osv$keypoint_control: [XREF] ost$keypoint_control;
   5943
o  6171
o  6172   VAR
o  6173     jmv$idle_dispatching_controls: [XREF] jmt$idle_dispatching_controls;
o  6174
   6194
   6195   VAR
   6196     jmv$ijl_entry_status_statistics: [XREF] jmt$ijl_entry_status_statistics;
   6197
o  6207
o  6208   VAR
```


NOS/VE Task Management : dispatcher

```
o  6209     jmv$job_counts: [XREF] jmt$job_counts;
o  6210
   6256   VAR
   6257     jmv$service_classes: [XREF, oss$mainframe_wired]
   6258           array [jmt$service_class_index] of ^jmt$service_class_entry;
   6259
o  6305
o  6306
o  6307   VAR
o  6308     jmv$null_ijl_ordinal: [XREF] jmt$ijl_ordinal;
o  6309
   6312
   6313   VAR
   6314     jmv$subsystem_priority_changes: [XREF] PACKED ARRAY [jmt$service_class_index] of boolean;
   6315
o  6318 [Pointer to the Active Segment Table - (AST).]
o  6319
o  6320   VAR
o  6321     mmv$ast_p: [XREF] ^mmt$active_segment_table;
o  6322
o  6325   VAR
o  6326     null_pva: 0 .. 0ffffffffffff(16),
o  6327     osv$system_family_name: [XDCL, #GATE] ost$name := '$SYSTEM                    ',
o  6328     jmv$system_core_id: [XDCL, #GATE] ost$name,
o  6329     tmv$cpu_execution_statistics: [XDCL, #GATE] tmt$cpu_execution_statistics,
o  6330     tmv$dct_priority_integer: integer,
o  6331     tmv$dispatch_priority_integer: [XDCL, #GATE] ARRAY [jmt$dispatching_priority] of integer,
o  6332     tmv$dispatching_controls: [XDCL] tmt$dispatching_controls,
o  6333     tmv$dispatching_control_sets: [XDCL, #GATE] tmt$dispatching_control_sets,
o  6334     tmv$dispatching_control_time: [XDCL] tmt$dispatching_pric_controls,
o  6335     tmv$dual_state_dispatch_prior: [XDCL, #GATE] tmt$dual_state_dispatch_prior :=
o  6336       [[1, 8], [1, 8], [2, 8], [2, 8], [3, 8], [3, 8], [4, 8], [4, 8], [5, 8], [5, 8],
o  6337        [6, 8], [6, 8], [7, 8], [7, 8]],
o  6338     tmv$io_wait_task_count: 0 .. tmc$maximum_ptl := 0,
o  6339     tmv$multiple_cpus_active: [XDCL, #GATE] boolean := FALSE,
o  6340     tmv$swapin_in_progress: [XDCL] integer := 0,
o  6341     tmv$time_to_call_dispatcher: [XDCL] integer := 0,
o  6342     tmv$timed_wait_not_queued: [XDCL, #GATE] integer := 600000000,
o  6343     tmv$timed_wait_queue: tmt$task_queue_link := [0, 0],
o  6344     tmv$total_task_count: [XDCL, #GATE] 0 .. tmc$maximum_ptl := 0,
o  6345     tmv$long_wait_swap_time: [XDCL, #GATE] integer := 6000000,
o  6346     tmv$long_wait_force_swap_time: [XDCL, #GATE] integer := 6000000,
o  6347     tmv$cycle_delay_time: [XDCL, #GATE] integer := 20000,
o  6348     mlv$c170_rqst_blk: [XREF] mlt$c170_rqst_blk,
o  6349     tmv$tables_initialized: [XDCL, #GATE] boolean := FALSE,
o  6350     syv$debug_control: [XDCL, #GATE] syt$debug_control,
o  6351     syv$all_jobs_selected_for_debug: [XDCL, #GATE] boolean := FALSE,
o  6352     tmv$ptl_lock: [XDCL] tmt$ptl_lock := [FALSE, 0],
o  6353     tmv$system_job_monitor_gtid: [XDCL] ost$global_task_id := [1, 1],
o  6354     job_time_zero: [XDCL] integer := 0,
o  6355     tmv$ptl_p: [XDCL, #GATE] ^tmt$primary_task_list,
o  6356     jmv$ajl_p: [XDCL, #GATE] ^jmt$active_job_list := NIL,
o  6357     jmv$ijl_p: [XDCL, #GATE] jmt$ijl_p := [NIL, 0, 0],
o  6358     jmv$ijle_size: [XDCL, #GATE] jmt$ijle_size := 0,
o  6359     jmv$swap_jobs_in_long_wait: [XDCL, #GATE] boolean := TRUE,
o  6360     jmv$system_ajl_ordinal: [XDCL, #GATE] jmt$ajl_ordinal := 0,
```

```
o  6361      jmv$system_ijl_ordinal: [XDCL, #GATE] jmt$ijl_ordinal := [0, 0],
o  6362      osv$special_aam_trap: [XDCL, #GATE] boolean := FALSE,
o  6363      pmv$quantum: [XDCL, #GATE] integer := 50000,
o  6364      tmv$dedicate_a_cpu_to_nos: [XDCL, #GATE] boolean := FALSE,
o  6365      tmv$subsystem_prior_threshold: [XDCL] 0 .. 0ff[16) := 5,
o  6366      tmv$dct: [XDCL, #GATE] tmt$dispatch_control_table;
o  6367
o  6368
o  6369 {Define types and variables for tracing CYCLE requests. Note that the code to trace CYCLE requests
o  6370 {is normally disabled and a special version of the system must be built to record information.
o  6371    ?VAR
o  6372      tmc$debug_cycle_requests: boolean := FALSE?;
o  6373
o  6374    TYPE
o  6375      cyctrace = record
o  6376        code: tmc$cycle_reason,
o  6377        status: tmt$task_status,
o  6378        p1: ^cell,
o  6379        p2: ^cell,
o  6380        gtid: ost$global_task_id,
o  6381        xtask: ost$task_index,
o  6382        time: 0 .. 0fffffffffff[16),
o  6383        p: ost$pva,
o  6384        utp: ^cell,
o  6385      recend;
o  6386
o  6387    ?IF tmc$debug_cycle_requests THEN
   6388
   6389      VAR
   6390        osv$debug: [XREF] integer,
   6391        tmv$cycle_trace: [XDCL, #GATE] array [0 .. 10001] of cyctrace,
   6392        ti: [STATIC] integer;
   6393
o  6394    ?IFEND
   6395
   6396    TYPE
   6397
   6398 { Priority_mask and dp_trick_conversion are types for the variable that is
   6399 { used to determine the priority to be considered for select_next_task.
   6400 { #unchecked_conversion is used to "pull off" the leftmost bit set.
   6401
   6402      priority_mask = packed record
   6403        fill: 0..0ff[16),
   6404        set_number: 0..0f[16),
   6405        dp: jmt$dispatching_priority,
   6406        fill2: 0..0fffffffffff[16),
   6407      recend,
   6408
   6409      dp_trick_conversion = record
   6410        case 0..1 of
   6411        = 0 =
   6412          r: real,
   6413        = 1 =
   6414          dp_mask: priority_mask,
   6415        casend,
   6416      recend;
```



```
   6417
   6418
```

NOS/VE Task Management : dispatcher

```
          6420
          6421   PROCEDURE [INLINE] tmp$set_lock (VAR lock: tmt$ptl_lock);
          6422
          6423     VAR
          6424       b: boolean,
          6425       bc: integer;
          6426
          6427     IF osv$cpus_logically_on > 1 THEN
          6428       bc := #read_register (osc$pr_base_constant);
          6429       IF lock.id <> bc  THEN
          6430         REPEAT
          6431          #TEST_SET (lock.locked, b);
          6432         UNTIL NOT b;
          6433         lock.id := bc;
          6434       ELSE
          6435         lock.count := lock.count + 1;
          6436       IFEND;
          6437     IFEND;
          6438
          6439 PROCEND tmp$set_lock;
          6440
        o 6443
        o 6444 PROCEDURE [INLINE] tmp$clear_lock (VAR lock: tmt$ptl_lock);
        o 6445
        o 6446     IF osv$cpus_logically_on > 1 THEN
        o 6447       IF lock.id <> #READ_REGISTER (osc$pr_base_constant) THEN
        o 6448         i#program_error; (Interlock failure - no message passed for performance reasons)
        o 6449       IFEND;
        o 6450       IF lock.count > 0 THEN
        o 6451         lock.count := lock.count - 1;
        o 6452       ELSE
        o 6453         lock.clear := 0;
        o 6454       IFEND;
        o 6455     IFEND;
        o 6456
        o 6457 PROCEND   tmp$clear_lock;
        o 6458
        o 6461
```


[XDCL, INLINE] tmp$check_taskid, tmp$check_taskid_with_lock_set

```
        o  6463
        o  6464   PROCEDURE [XDCL, INLINE] tmp$check_taskid
        4  6465     (     taskid: ost$global_task_id;
        4  6466           option: tmt$option;
        4  6467      VAR status: syt$monitor_status);
        4  6468
        4  6469
        4  6470 (
        4  6471 (  The purpose of this procedure is to check for a valid taskid.
        4  6472 (  Depending on the option selected, this module either halts the
        4  6473 (  system or returns the status in case an error condition is
        4  6474 (  encountered.
        4  6475 (  The callers of this procedure DO NOT set the ptl lock.  This
        4  6476 (  procedure should be used by callers outside of the 'TM' decks.
        4  6477 (  'TM' callers should use TMP$CHECK_TASKID_WITH_LOCK_SET.
        4  6478 (
        4  6479 (    TMP$CHECK_TASKID (TASKID, OPTION, STATUS)
        4  6480 (
        4  6481 (  TASKID: (INPUT) This parameter specifies the taskid.
        4  6482 (
        4  6483 (  OPTION: (INPUT) This parameter specifies the option to
        4  6484 (                  halt the system or to return an error status.
        4  6485 (
        4  6486 (  STATUS: (OUTPUT) This parameter specifies the error status.
        4  6487 (
        4  6488
        4  6489     status.normal := TRUE;
        4  6490     tmp$set_lock (tmv$ptl_lock);
       4A  6491     IF (taskid = tmv$null_global_task_id) OR (taskid.index > UPPERBOUND (tmv$ptl_p^)) OR
       8C  6492           (tmv$ptl_p^ [taskid.index].sequence_number <> taskid.seqno) OR
       8C  6493           (tmv$ptl_p^ [taskid.index].status = tmc$ts_null) THEN
       8C  6494       tmp$clear_lock (tmv$ptl_lock);
       C2  6495       IF option = tmc$opt_return THEN
       CA  6496         mtp$set_status_abnormal ('TM', tme$invalid_global_taskid, status);
       CA  6497         RETURN;
       E2  6498       ELSE
       E2  6499         mtp$error_stop ('TM01 - taskid error');
      104  6500       IFEND;
      104  6501     IFEND;
      104  6502     tmp$clear_lock (tmv$ptl_lock);
      138  6503   PROCEND tmp$check_taskid;


        0  6505   PROCEDURE [XDCL, INLINE] tmp$check_taskid_with_lock_set
        4  6506     (     taskid: ost$global_task_id;
        4  6507           option: tmt$option;
        4  6508      VAR status: syt$monitor_status);
        4  6509
        4  6510 (
        4  6511 (  The purpose of this procedure is to check for a valid taskid.
        4  6512 (  Depending on the option selected, this module either halts the
        4  6513 (  system or returns the status in case an error condition is
        4  6514 (  encountered.
        4  6515 (  The callers of this procedure MUST set the ptl lock.  Callers
        4  6516 (  outside of the 'TM' decks should use TMP$CHECK_TASKID, which
```

[XDCL, INLINE] tmp$check_taskid, tmp$check_taskid_with_lock_set

```
4  6517 {  does the ptl lock management.
4  6518 {
4  6519 {     TMP$CHECK_TASKID_WITH_LOCK_SET (TASKID, OPTION, STATUS)
4  6520 {
4  6521 {  TASKID: (INPUT) This parameter specifies the taskid.
4  6522 {
4  6523 {  OPTION: (INPUT) This parameter specifies the option to
4  6524 {                  halt the system or to return an error status.
4  6525 {
4  6526 {  STATUS: (OUTPUT) This parameter specifies the error status.
4  6527 {
4  6528
4  6529      status.normal := TRUE;
4  6530      IF (taskid = tmv$null_global_task_id) OR (taskid.index > UPPERBOUND (tmv$ptl_p^)) OR
50 6531            (tmv$ptl_p^ [taskid.index].sequence_number <> taskid.seqno) OR
50 6532            (tmv$ptl_p^ [taskid.index].status = tmc$ts_null) THEN
50 6533        IF option = tmc$opt_return THEN
58 6534          mtp$set_status_abnormal ('TM', tme$invalid_global_taskid, status);
58 6535          RETURN;
6C 6536        ELSE
6C 6537          mtp$error_stop ('TM01 - taskid error');
8C 6538        IFEND;
8C 6539      IFEND;
8C 6540    PROCEND tmp$check_taskid_with_lock_set;
 0 6541
```


[INLINE] tmp$stop_if_bad_taskid

```
0  6543
0  6544    PROCEDURE [INLINE] tmp$stop_if_bad_taskid
0  6545      (     taskid: ost$global_task_id);
0  6546
0  6547 { NOTE: The caller of this procedure MUST set the PTL lock.
0  6548
0  6549      IF (taskid = tmv$null_global_task_id) OR (taskid.index > UPPERBOUND (tmv$ptl_p^)) OR
0  6550            (tmv$ptl_p^ [taskid.index].sequence_number <> taskid.seqno) OR
0  6551            (tmv$ptl_p^ [taskid.index].status = tmc$ts_null) THEN
0  6552        mtp$error_stop ('TM01 - taskid error');
0  6553      IFEND;
0  6554
0  6555    PROCEND tmp$stop_if_bad_taskid;
```

TMP$OBTAIN_IJL_ORDINAL_FOR_PTL

```
o  6557
o  6558   PROCEDURE [XDCL, INLINE] tmp$obtain_ijl_ordinal_from_ptl
4  6559     (    global_task_id: ost$global_task_id;
4  6560      VAR ijl_ordinal: jmt$ijl_ordinal);
4  6561
4  6562     ijl_ordinal := tmv$ptl_p^ [global_task_id.index].ijl_ordinal;
4  6563
4  6564   PROCEND tmp$obtain_ijl_ordinal_from_ptl;
```


ASSIGN_PTL, FREE_PTL

```
o  6566
o  6567   PROCEDURE [INLINE]  tmp$assign_ptl
o  6568     (    xcb_p: ^ost$execution_control_block;
o  6569          ijl_ordinal: jmt$ijl_ordinal;
o  6570      VAR taskid: ost$global_task_id;
o  6571      VAR status: syt$monitor_status);
o  6572
o  6573 {
o  6574 {  The purpose of this procedure is to assign PTL entries from
o  6575 {  the free queue.  An error status is returned if the PTL is full.
o  6576 {
o  6577 {    ASSIGN_PTL (XCB_P, JOB_ID, TASKID, STATUS)
o  6578 {
o  6579 {  XCB_P : (INPUT) This parameter specifies the pointer to XCB.
o  6580 {
o  6581 {  JOB_ID : (INPUT) This parameter specifies the ordinal of the
o  6582 {                   active job list entry for the job.
o  6583 {
o  6584 {  TASKID : (OUTPUT) This parameter specifies the taskid assigned
o  6585 {                    to the task.
o  6586 {
o  6587 {  STATUS : (OUTPUT) This parameter specifies the error status.
o  6588 {
o  6589 {  NOTE : Anyone calling this procedure must have tmv$ptl_lock set.
o  6590 {
o  6591
o  6592     VAR
o  6593       ijle_p: ^jmt$initiated_job_list_entry,
o  6594       next_avail_ptlo: ost$task_index,
o  6595       ptl_p: ^tmt$primary_task_list_entry;
o  6596
o  6597
o  6598
o  6599 {Check if PTL space is available.
o  6600
o  6601     status.normal := TRUE;
o  6602     next_avail_ptlo := tmv$dct [jmc$null_dispatching_priority].queue_head;
o  6603     IF next_avail_ptlo = 0 THEN
o  6604       mtp$set_status_abnormal ('TM', tme$ptl_full, status);
o  6605       RETURN;
o  6606     IFEND;
o  6607     ptl_p := ^tmv$ptl_p^ [next_avail_ptlo];
o  6608     tmv$dct [jmc$null_dispatching_priority].queue_head := ptl_p^.ptl_thread;
o  6609
o  6610 { If the job is selected for system breakpoints, set the flag for the task to set up debug registers.
o  6611
o  6612     jmp$get_ijle_p (ijl_ordinal, ijle_p);
o  6613     IF ijle_p^.system_breakpoint_selected THEN
o  6614       ptl_p^.monitor_flags := $syt$monitor_flags [syc$mf_system_debugger];
o  6615     ELSE
o  6616       ptl_p^.monitor_flags := $syt$monitor_flags [];
o  6617     IFEND;
o  6618
o  6619 {Initialize the PTL entry.
o  6620
o  6621     xcb_p^.keypoint_enable := FALSE;
```

ASSIGN_PTL, FREE_PTL

```
o  6622      xcb_p^.keypoint_register_enable := FALSE;
o  6623      ptl_p^.status := tmc$ts_ready;
o  6624      ptl_p^.ijl_ordinal := ijl_ordinal;
o  6625      ptl_p^.xcb_offset := #OFFSET (xcb_p);
o  6626      ptl_p^.dispatching_priority := xcb_p^.dispatching_priority;
o  6627      ptl_p^.sequence_number := ptl_p^.sequence_number MOD 255 + 1;
o  6628      ptl_p^.system_flags := $tmt$system_flags [];
o  6629      ptl_p^.idle_status := tmc$is_not_idled;
o  6630      ptl_p^.queue_link.head := 0;
o  6631      ptl_p^.queue_link.tail := 0;
o  6632      ptl_p^.end_of_wait_time := 0;
o  6633      ptl_p^.readying_task_priority := 0;
o  6634      taskid.index := next_avail_ptlo;
o  6635      taskid.seqno := ptl_p^.sequence_number;
o  6636
o  6637    PROCEND tmp$assign_ptl;


o  6639
o  6640 {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
o  6641 {  The caller must also call the procedure remove_ijl to remove the task from the ijl_thread,
o  6642 {  which links all tasks of a job, if the task being freed is NOT the job monitor task.
o  6643 {  If all tasks are being freed, the caller can zero out the ijl_thread.
o  6644
o  6645    PROCEDURE [INLINE] free_ptl
o  6646      (     ptlo: ost$task_index);
o  6647
o  6648      tmv$ptl_p^ [ptlo].status := tmc$ts_null;
o  6649      tmv$ptl_p^ [ptlo].dispatching_priority := jmc$null_dispatching_priority;
o  6650      tmv$ptl_p^ [ptlo].ptl_thread := 0;
o  6651      IF tmv$dct [jmc$null_dispatching_priority].queue_head = 0 THEN
o  6652        tmv$dct [jmc$null_dispatching_priority].queue_head := ptlo;
o  6653      ELSE
o  6654        tmv$ptl_p^ [tmv$dct [jmc$null_dispatching_priority].queue_tail].ptl_thread := ptlo;
o  6655      IFEND;
o  6656      tmv$dct [jmc$null_dispatching_priority].queue_tail := ptlo;
o  6657    PROCEND free_ptl;
```


REMOVE_TASK_FROM_DCT

```
o  6659
o  6660    PROCEDURE  [INLINE]  tmp$remove_task_from_dct
o  6661      (     ptlo: ost$task_index);
o  6662
o  6663 {
o  6664 {  The purpose of this procedure is to remove the current task's
o  6665 {  PTL entry from the DCT thread.
o  6666 {
o  6667 {    REMOVE_TASK_FROM_DCT (PTLO)
o  6668 {
o  6669 {  PTLO : (INPUT) This parameter specifies the PTL index of the
o  6670 {                   specified task.
o  6671 {
o  6672 {  NOTE : Anyone calling this procedure must have tmv$ptl_lock set.
o  6673 {
o  6674
o  6675      VAR
o  6676        dcte: tmt$dct_entry,
o  6677        scan_ptlo: ost$task_index,
o  6678        save_ptlo: ost$task_index,
o  6679        ptle_p: ^tmt$primary_task_list_entry;
o  6680
o  6681      ptle_p := ^tmv$ptl_p^ [ptlo];
o  6682      dcte := tmv$dct [ptle_p^.dispatching_priority];
o  6683
o  6684      IF dcte.queue_tail = dcte.queue_head THEN
o  6685        IF dcte.queue_head <> ptlo THEN
o  6686          mtp$error_stop ('TM--remove task from empty DCT');
o  6687        IFEND;
o  6688        dcte.queue_tail := 0;
o  6689        dcte.queue_head := 0;
o  6690        dcte.minor_priority := 0;
o  6691        dcte.major_priority := 0;
o  6692        tmv$dispatching_control_sets.ready_tasks := tmv$dispatching_control_sets.ready_tasks -
o  6693              $jmt$dispatching_priority_set [jmc$dp_conversion - ptle_p^.dispatching_priority];
o  6694        tmp$calculate_dct_priority_int;
o  6695      ELSEIF ptlo = dcte.queue_head THEN
o  6696        dcte.queue_head := ptle_p^.ptl_thread;
o  6697        IF ptlo = dcte.major_priority THEN
o  6698          dcte.major_priority := ptle_p^.ptl_thread;
o  6699        IFEND;
o  6700        IF ptlo = dcte.minor_priority THEN
o  6701          dcte.minor_priority := ptle_p^.ptl_thread;
o  6702        IFEND;
o  6703      ELSE
o  6704        scan_ptlo := dcte.queue_head;
o  6705        WHILE scan_ptlo <> ptlo DO
o  6706          save_ptlo := scan_ptlo;
o  6707          scan_ptlo := tmv$ptl_p^ [scan_ptlo].ptl_thread;
o  6708          IF scan_ptlo = 0 THEN
o  6709            mtp$error_stop ('TM-cant find ptlo to remove from DCT');
o  6710          IFEND;
o  6711        WHILEND;
o  6712        tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [ptlo].ptl_thread;
o  6713        IF ptlo = dcte.queue_tail THEN
o  6714          dcte.queue_tail := save_ptlo;
```

REMOVE_TASK_FROM_DCT

```
o  6715        IFEND;
o  6716        IF ptlo = dcte.major_priority THEN
o  6717          dcte.major_priority := save_ptlo;
o  6718        IFEND;
o  6719        IF ptlo = dcte.minor_priority THEN
o  6720          dcte.minor_priority := save_ptlo;
o  6721        IFEND;
o  6722      IFEND;
o  6723      ptle_p^.ptl_thread := 0;
o  6724      tmv$dct [ptle_p^.dispatching_priority] := dcte;
o  6725
o  6726    PROCEND tmp$remove_task_from_dct;
o  6727
```


REMOVE_TASK_FROM_FREE_QUEUE

```
o  6729
o  6730    PROCEDURE [INLINE] remove_task_from_free_queue
o  6731      (     ptlo: ost$task_index);
o  6732
o  6733 {
o  6734 {  The purpose of this procedure is to remove a task's PTL entry
o  6735 {  from the free DCT thread.
o  6736 {
o  6737 {    REMOVE_TASK_FROM_FREE_QUEUE (PTLO)
o  6738 {
o  6739 {  PTLO : (INPUT) This parameter specifies the PTL index of the
o  6740 {                  specified task.
o  6741 {
o  6742 {  NOTE : Anyone calling this procedure must have tmv$ptl_lock set.
o  6743 {
o  6744
o  6745      VAR
o  6746        dcte: tmt$dct_entry,
o  6747        save_ptlo: ost$task_index,
o  6748        scan_ptlo: ost$task_index,
o  6749        ptle_p: ^tmt$primary_task_list_entry;
o  6750
o  6751      ptle_p := ^tmv$ptl_p^ [ptlo];
o  6752      dcte := tmv$dct [jmc$null_dispatching_priority];
o  6753
o  6754      IF dcte.queue_tail = dcte.queue_head THEN
o  6755        IF dcte.queue_head <> ptlo THEN
o  6756          mtp$error_stop ('TM--task not in free queue');
o  6757        IFEND;
o  6758        dcte.queue_head := 0;
o  6759        dcte.queue_tail := 0;
o  6760      ELSEIF ptlo = dcte.queue_head THEN
o  6761        dcte.queue_head := ptle_p^.ptl_thread;
o  6762      ELSE
o  6763        scan_ptlo := dcte.queue_head;
o  6764        WHILE (scan_ptlo <> ptlo) AND (scan_ptlo <> 0) DO
o  6765          save_ptlo := scan_ptlo;
o  6766          scan_ptlo := tmv$ptl_p^ [scan_ptlo].ptl_thread;
o  6767        WHILEND;
o  6768        IF scan_ptlo = 0 THEN
o  6769          mtp$error_stop ('TM--task not in free queue');
o  6770        IFEND;
o  6771        tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [ptlo].ptl_thread;
o  6772        IF ptlo = dcte.queue_tail THEN
o  6773          dcte.queue_tail := save_ptlo;
o  6774        IFEND;
o  6775      IFEND;
o  6776      tmv$dct [jmc$null_dispatching_priority] := dcte;
o  6777
o  6778    PROCEND remove_task_from_free_queue;
o  6779
```

[INLINE] tmp$dct_ready_task

```
o  6781
o  6782 { PURPOSE:
o  6783 {   The purpose of this procedure is to pre-select a CPU for a ready task
o  6784 {   to execute on, or place the ready task into the dispatch_control_table (DCT).
o  6785 {
o  6786 { DESIGN:
o  6787 {   Integer representation of dispatching priorities:
o  6788 {     To accommodate dispatching allocation (guaranteeing minimum and maximum
o  6789 {     percentages of the total CPU time to specified user dispatching priorities),
o  6790 {     each dispatching priority falls into one of three sets.  The sets are defined
o  6791 {     in one word of memory in the following order:
o  6792 {
o  6793 {     BYTE:     63.....................................................................0
o  6794 {     PRIORITY:                  15    ..      0 15     ..      0 15     ..      0
o  6795 {
o  6796 {               ,-----------------,-----------------,-----------------,-----------------,
o  6797 {               |                 |                 |                 |                 |
o  6798 {               '-----------------'-----------------'-----------------'-----------------'
o  6799 {                                       SET 3             SET 2             SET 1
o  6800 {
o  6801 {     SET 1:  Tasks of priorities in this set have exceeded their maximum dispatching
o  6802 {             priority allocation.
o  6803 {     SET 2:  Tasks of priorities in this set have achieved their minimum dispatching
o  6804 {             priority allocation, but have not exceeded their maximum allocation.
o  6805 {     SET 3:  Tasks of priorities in this set have not yet achieved their minimum
o  6806 {             dispatching allocation.
o  6807 {
o  6808 {     When the word containing the dispatching priority sets is normalized and
o  6809 {     converted to an integer, any SET 3 task has a higher dispatching priority integer
o  6810 {     than any SET 2 task, which in turn has a higher dispatching priority integer
o  6811 {     than any SET 1 task.  Thus, a SET 3-P4 task has a higher dispatching priority
o  6812 {     integer than a SET 2-P8 task.
o  6813 {
o  6814 {     NOTE:  Sub-system and system priorities (P9..P14) are always placed in SET 3, so
o  6815 {            they will always have a higher dispatching priority integer value than
o  6816 {            any user dispatching priority.
o  6817 {
o  6818 {   Task pre-selection or placement in the DCT:
o  6819 {     Task pre-selection will be attempted only if the job with the ready task allows
o  6820 {     multiprocessing or has no other tasks currently executing.  If the ready task
o  6821 {     is going through task_switch and is still ready (its time slice has expired),
o  6822 {     pre-selection will be attempted only if the job allows multiprocessing or has
o  6823 {     no other ready tasks.  (This prevents a non-multiprocessing job with a CPU bound
o  6824 {     task from shutting out other tasks of the job.)
o  6825 {
o  6826 {     The ready task will be pre-selected to run on the first idle CPU that is found,
o  6827 {     or on the the lowest dispatching priority (integer) CPU currently executing, if
o  6828 {     the ready task has a higher dispatching priority integer.  If the CPU that is
o  6829 {     pre-selected is currently executing another task, that CPU will be interrupted.
o  6830 {
o  6831 {     If the ready task cannot be pre-selected to execute on a CPU, the task is inserted
o  6832 {     into the DCT.
o  6833 {
o  6834 {   The dispatch_control_table (DCT):
o  6835 {     The DCT is an array of singularly linked lists. There is a list for each
o  6836 {     dispatching priority.  There are four different pointers into each list:
o  6837 {     the head pointer, the tail pointer, the minor priority pointer, and the
```


[INLINE] tmp$dct_ready_task

```
o  6837 {     major priority pointer.
o  6838 {
o  6839 {     Each job class has a major and a minor timeslice. The major and minor
o  6840 {     timeslice of a job class can be modified by using the change_class_attribute
o  6841 {     command. When a task is initially dispatched, it is placed after the minor
o  6842 {     priority pointer in the DCT. "Ready tasks" are placed after the minor
o  6843 {     priority pointer as long as the minor timeslice in the tasks XCB is greater
o  6844 {     than one_eighth of the job class minor timeslice. Both the minor and the
o  6845 {     major timeslice in the XCB of a task are decremented by the amount of time that
o  6846 {     the task uses the CPU. Once the minor timeslice in the XCB is zero, the task
o  6847 {     is placed after the major priority pointer in the dispatch control table.
o  6848 {     The minor timeslice in the XCB is also reset. When a task finally uses all
o  6849 {     of its major timeslice, it is next dispatched at the tail of the DCT. Both
o  6850 {     the major and minor timeslice in the XCB are then reset.
o  6851 {
o  6852 {     |----------|
o  6853 {     |   HEAD   | ------------->  TASK
o  6854 {     |----------|                 TASK          Example of a DCT with five
o  6855 {     |  MINOR   | ------------->  TASK          tasks waiting to be dispatched.
o  6856 {     |----------|
o  6857 {     |  MAJOR   | ------------->  TASK
o  6858 {     |----------|
o  6859 {     |   TAIL   | ------------->  TASK
o  6860 {     |----------|
o  6861
o  6862
o  6863   PROCEDURE [INLINE] tmp$dct_ready_task
o  6864     (     xcb_p: ^ost$execution_control_block;
o  6865           ijle_p: ^jmt$initiated_job_list_entry;
o  6866           ptlo: ost$task_index;
o  6867           attempt_preselection: boolean);
o  6868
o  6869     VAR
o  6870       cst_index: integer,
o  6871       cst_p: ^ost$cpu_state_table,
o  6872       dcte: tmt$dct_entry,
o  6873       dct_placement: (minor_timeslice_insert, major_timeslice_insert, tail_timeslice_insert),
o  6874       insert_ptle_p: ^tmt$primary_task_list_entry,
o  6875       insert_ijle_p: ^jmt$initiated_job_list_entry,
o  6876       low_priority: integer,
o  6877       low_pric_csti: integer,
o  6878       ptle_p: ^tmt$primary_task_list_entry,
o  6879       timeslice: jmt$time_slice_values;
o  6880
o  6881     ptle_p := ^tmv$ptl_p^ [ptlo];
o  6882
o  6883     ?IF NOT debug THEN
o  6884       timeslice := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
o  6885           dispatching_control [ijle_p^.dispatching_control.dispatching_control_index].dispatching_timeslice;
o  6886     ?ELSE
   6887       timeslice.minor := 50;
   6888       timeslice.major := 50;
o  6889     ?IFEND;
o  6890
o  6891 { Determine where the task is to be positioned in the DISPATCH_CONTROL_TABLE.
o  6892
```

[INLINE] tmp$dct_ready_task

```
o  6893      IF xcb_p^.timeslice.minor > (timeslice.minor DIV 8) THEN
o  6894        dct_placement := minor_timeslice_insert;
o  6895      ELSEIF xcb_p^.timeslice.major > (timeslice.minor DIV 8) THEN
o  6896        dct_placement := major_timeslice_insert;
o  6897        xcb_p^.timeslice.minor := timeslice.minor;
o  6898      ELSE
o  6899        dct_placement := tail_timeslice_insert;
o  6900        xcb_p^.timeslice.major := timeslice.major;
o  6901        xcb_p^.timeslice.minor := timeslice.minor;
o  6902      IFEND;
o  6903
o  6904 { If a task has subsystem table locks set for more than the subsystem lock threshold, then
o  6905 { lower the tasks priority. If the task received a READY request while it had subsystem
o  6906 { locks set, the priority is lowered to the greater of the task's or the READYING task's
o  6907 { dispatching priority.
o  6908
o  6909      IF (xcb_p^.system_table_lock_count > 0) THEN
o  6910        IF (xcb_p^.system_table_lock_count < osc$system_table_lock_set) THEN
o  6911          IF xcb_p^.subsystem_lock_priority_count >= tmv$subsystem_prior_threshold THEN
o  6912            IF (xcb_p^.dispatching_priority < ptle_p^.readying_task_priority) THEN
o  6913              ptle_p^.dispatching_priority := ptle_p^.readying_task_priority;
o  6914            ELSE
o  6915              ptle_p^.dispatching_priority := xcb_p^.dispatching_priority;
o  6916            IFEND;
o  6917            xcb_p^.subsystem_give_up_cpu := TRUE;
o  6918          ELSE
o  6919            IF ptle_p^.dispatching_priority < jmc$prior_subsystem_tbls_locked THEN
o  6920              ptle_p^.dispatching_priority := jmc$prior_subsystem_tbls_locked;
o  6921            IFEND;
o  6922            IF dct_placement <> minor_timeslice_insert THEN
o  6923              xcb_p^.subsystem_lock_priority_count := xcb_p^.subsystem_lock_priority_count + 1;
o  6924            IFEND;
o  6925            xcb_p^.subsystem_give_up_cpu := TRUE;
o  6926          IFEND;
o  6927        ELSE
o  6928          xcb_p^.system_give_up_cpu := TRUE;
o  6929          IF ptle_p^.dispatching_priority < jmc$prior_system_tbls_locked THEN
o  6930            ptle_p^.dispatching_priority := jmc$prior_system_tbls_locked;
o  6931          IFEND;
o  6932        IFEND;
o  6933      IFEND;
o  6934
o  6935 { Determine if the priority is high enough to preempt a task that is currently executing.
o  6936 { Send an interrupt to the processor executing the lowest priority task (or an idle processor).
o  6937
o  6938      IF attempt_preselection THEN
o  6939
o  6940        low_priority := tmv$dispatch_priority_integer [ptle_p^.dispatching_priority];
o  6941        mtp$cst_p (cst_p);
o  6942        cst_index := osv$cpus_physically_configured - 1;
o  6943        low_prio_csti := -1;
o  6944        /preselect_loop/
o  6945        BEGIN
o  6946        REPEAT
o  6947          IF (mtv$cst0 [cst_index].dispatching_priority_integer < low_priority)
o  6948              AND (cst_index IN xcb_p^.processor_selections) THEN
```


[INLINE] tmp$dct_ready_task

```
o  6949            IF tmv$dedicate_a_cpu_to_nos AND (cst_index = 0) THEN
o  6950              EXIT /preselect_loop/;
o  6951            IFEND;
o  6952            low_prio_csti := cst_index;
o  6953            low_priority := mtv$cst0 [cst_index].dispatching_priority_integer;
o  6954          IFEND;
o  6955          cst_index := cst_index - 1;
o  6956        UNTIL (cst_index < 0) OR (low_priority = 0);
o  6957        END  /preselect_loop/;
o  6958
o  6959        IF low_prio_csti >= 0 THEN
o  6960          IF (mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch <> 0) THEN
o  6961
o  6962 { A task has already been pre-selected to run on the pre-selected processor.  The task must be inserted
o  6963 { at the head of its priority DCT queue.
o  6964
o  6965            insert_ptle_p := ^tmv$ptl_p^ [mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch];
o  6966            jmp$get_ijle_p (insert_ptle_p^.ijl_ordinal, insert_ijle_p);
o  6967            insert_ijle_p^.executing_task_count := insert_ijle_p^.executing_task_count - 1;
o  6968            insert_ptle_p^.status := tmc$ts_ready;
o  6969            dcte := tmv$dct [insert_ptle_p^.dispatching_priority];
o  6970            IF dcte.queue_head = 0 THEN
o  6971              insert_ptle_p^.ptl_thread := 0;
o  6972              dcte.queue_head := mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch;
o  6973              dcte.queue_tail := mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch;
o  6974              dcte.minor_priority := mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch;
o  6975              dcte.major_priority := mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch;
o  6976              tmv$dispatching_control_sets.ready_tasks := tmv$dispatching_control_sets.ready_tasks +
o  6977                    $jmt$dispatching_priority_set [jmc$dp_conversion -
o  6978                    insert_ptle_p^.dispatching_priority];
o  6979              tmp$calculate_dct_priority_int;
o  6980            ELSE
o  6981              insert_ptle_p^.ptl_thread := dcte.queue_head;
o  6982              dcte.queue_head := mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch;
o  6983            IFEND;
o  6984            tmv$dct [insert_ptle_p^.dispatching_priority] := dcte;
o  6985          IFEND;
o  6986
o  6987          mtv$cst0 [low_prio_csti].next_ptlo_to_dispatch := ptlo;
o  6988          mtv$cst0 [low_prio_csti].dispatching_priority_integer := tmv$dispatch_priority_integer
o  6989                [ptle_p^.dispatching_priority];
o  6990          mtv$cst0 [low_prio_csti].dispatch_control.call_dispatcher := TRUE;
o  6991          ptle_p^.status := tmc$ts_ready_and_selected;
o  6992          ijle_p^.executing_task_count := ijle_p^.executing_task_count + 1;
o  6993          IF (low_prio_csti <> cst_p^.cst_index) AND ((low_priority <> 0 ) OR
o  6994                (mtv$cst0 [low_prio_csti].dual_state_jps <> 0)) THEN
o  6995            mtp$interrupt_processor (mtv$cst0 [low_prio_csti].memory_port_mask);
o  6996          IFEND;
o  6997          RETURN;
o  6998        IFEND; {  low_prio_csti >= 0 }
o  6999      IFEND; { attempt_preselection }
o  7000
o  7001 { If no processor is executing a task with a lower priority then the task being readied must be
o  7002 { placed in the DCT queue.
o  7003
o  7004      dcte := tmv$dct [ptle_p^.dispatching_priority];
```

[INLINE] tmp$dct_ready_task

```
o  7005
o  7006      IF dcte.queue_head = 0 THEN
o  7007        ptle_p^.ptl_thread := 0;
o  7008        dcte.queue_head := ptlo;
o  7009        dcte.queue_tail := ptlo;
o  7010        dcte.minor_priority := ptlo;
o  7011        dcte.major_priority := ptlo;
o  7012        tmv$dispatching_control_sets.ready_tasks := tmv$dispatching_control_sets.ready_tasks +
o  7013              $jmt$dispatching_priority_set [jmc$dp_conversion - ptle_p^.dispatching_priority];
o  7014        tmp$calculate_dct_priority_int;
o  7015        tmv$dct [ptle_p^.dispatching_priority] := dcte;
o  7016        RETURN;
o  7017      IFEND;  { dcte.queue_head = 0 }
o  7018
o  7019      IF dct_placement = minor_timeslice_insert THEN
o  7020        ptle_p^.ptl_thread := tmv$ptl_p^ [dcte.minor_priority].ptl_thread;
o  7021        tmv$ptl_p^ [dcte.minor_priority].ptl_thread := ptlo;
o  7022
o  7023 { The following is required to insure that the major pointer
o  7024 { is never higher than the minor pointer in the DCT.
o  7025
o  7026        IF dcte.minor_priority = dcte.major_priority THEN
o  7027          dcte.major_priority := ptlo;
o  7028        IFEND;
o  7029        dcte.minor_priority := ptlo;
o  7030        IF ptle_p^.ptl_thread = 0 THEN
o  7031          dcte.queue_tail := ptlo;
o  7032        IFEND;
o  7033      ELSEIF dct_placement = major_timeslice_insert THEN
o  7034        ptle_p^.ptl_thread := tmv$ptl_p^ [dcte.major_priority].ptl_thread;
o  7035        tmv$ptl_p^ [dcte.major_priority].ptl_thread := ptlo;
o  7036        dcte.major_priority := ptlo;
o  7037        IF ptle_p^.ptl_thread = 0 THEN
o  7038          dcte.queue_tail := ptlo;
o  7039        IFEND;
o  7040      ELSEIF dct_placement = tail_timeslice_insert THEN
o  7041        tmv$ptl_p^ [dcte.queue_tail].ptl_thread := ptlo;
o  7042        ptle_p^.ptl_thread := 0;
o  7043        dcte.queue_tail := ptlo;
o  7044      ELSE
o  7045        mtp$error_stop (' TM99--Illegal insert ');
o  7046      IFEND;
o  7047
o  7048    { If any task in a job has specified a relative task priority, all of
o  7049    { the tasks in the job are subject to the relative priority algorithms.
o  7050
o  7051      IF (ijle_p^.relative_priority_enabled) AND (ijle_p^.statistics.
o  7052           ready_task_count > 1) THEN
o  7053        IF dct_placement = minor_timeslice_insert THEN
o  7054          relative_insert_minor_dct (xcb_p^.relative_task_priority, ptlo, ijle_p, dcte);
o  7055        ELSEIF dct_placement = major_timeslice_insert THEN
o  7056          relative_insert_major_dct (xcb_p^.relative_task_priority, ptlo, ijle_p, dcte);
o  7057        ELSEIF dct_placement = tail_timeslice_insert THEN
o  7058          relative_insert_tail_dct (xcb_p^.relative_task_priority, ptlo, ijle_p, dcte);
o  7059        IFEND;
o  7060      IFEND;
```


[INLINE] tmp$dct_ready_task

```
o  7061
o  7062      tmv$dct [ptle_p^.dispatching_priority] := dcte;
o  7063
o  7064    PROCEND tmp$dct_ready_task;
o  7065
```

```
o  7067
o  7068   PROCEDURE  [INLINE] relative_insert_minor_dct
o  7069     (    relative_priority: 0 .. 255;
o  7070          ptlo: ost$task_index;
o  7071          ijl_p: ^jmt$initiated_job_list_entry;
o  7072      VAR dcte: tmt$dct_entry);
o  7073
o  7074 { The following procedure is responsible for making sure a task which was
o  7075 { inserted in the MINOR timeslice position is corrected prioritized with the
o  7076 { other tasks of the same job. This job is currently running with relative
o  7077 { prioritization enabled. The position of tasks of other jobs can only be
o  7078 { improved, they will never regress.
o  7079
o  7080      VAR
o  7081        insert_ptlo: ost$task_index,
o  7082        next_search_ptlo: ost$task_index,
o  7083        save_ptlo: ost$task_index,
o  7084        search_ptlo: ost$task_index,
o  7085        search_xcb: ^ost$execution_control_block;
o  7086
o  7087   { Insert_ptlo is the most recently moved ptlo.
o  7088   { Search_ptlo is the task which we are currently testing for
o  7089   { possible movement.
o  7090   { Save_ptlo is the last task which was NOT moved.
o  7091
o  7092      insert_ptlo := ptlo;
o  7093      search_ptlo := dcte.queue_head;
o  7094      save_ptlo := 0;
o  7095
o  7096   { The task has been positioned in the DCT. Now, all the tasks of that
o  7097   { job (currently in DCT) must be prioritized. The task has been placed
o  7098   { in the minor priority position. Tasks above that point are checked to
o  7099   { determine if any of them have to be moved below the new task. If the
o  7100   { task is in the same job, and has a lower relative priority, the task is
o  7101   { moved below the new task.
o  7102
o  7103      WHILE (search_ptlo <> ptlo) DO
o  7104        next_search_ptlo := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7105        IF (tmv$ptl_p^ [search_ptlo].ijl_ordinal = tmv$ptl_p^ [ptlo].ijl_ordinal) THEN
o  7106          search_xcb := #ADDRESS (1, ijl_p^.ajl_ordinal + mtc$job_fixed_segment,
o  7107            tmv$ptl_p^ [search_ptlo].xcb_offset);
o  7108          IF (search_xcb^.relative_task_priority < relative_priority) THEN
o  7109            IF search_ptlo = dcte.queue_head THEN
o  7110              dcte.queue_head := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7111            ELSE
o  7112              tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7113            IFEND;
o  7114            tmv$ptl_p^ [search_ptlo].ptl_thread := tmv$ptl_p^ [insert_ptlo].ptl_thread;
o  7115            tmv$ptl_p^ [insert_ptlo].ptl_thread := search_ptlo;
o  7116            insert_ptlo := search_ptlo;
o  7117          ELSE
o  7118            save_ptlo := search_ptlo;
o  7119          IFEND;
o  7120        ELSE
o  7121          save_ptlo := search_ptlo;
o  7122        IFEND;
```



```
o  7123        search_ptlo := next_search_ptlo;
o  7124      WHILEND;
o  7125
o  7126      IF insert_ptlo <> ptlo THEN
o  7127
o  7128   { One or more tasks have been moved below the new task.
o  7129
o  7130        IF dcte.minor_priority = dcte.major_priority THEN
o  7131          dcte.major_priority := insert_ptlo;
o  7132        IFEND;
o  7133        IF tmv$ptl_p^ [insert_ptlo].ptl_thread = 0 THEN
o  7134          dcte.queue_tail := insert_ptlo;
o  7135        IFEND;
o  7136        dcte.minor_priority := insert_ptlo;
o  7137      ELSE
o  7138
o  7139 { There were not any tasks moved below the new task. We must now check to
o  7140 { see if there are any tasks below the new task, which have a relative priority
o  7141 { higher than the new task. A search is made of all of the tasks below the new
o  7142 { task, the new task is moved below the last task (of the same job) which
o  7143 { has a higher relative priority than the new task.
o  7144
o  7145        insert_ptlo := 0;
o  7146        search_ptlo := tmv$ptl_p^ [ptlo].ptl_thread;
o  7147        WHILE search_ptlo <> 0 DO
o  7148          IF (tmv$ptl_p^ [search_ptlo].ijl_ordinal = tmv$ptl_p^ [ptlo].ijl_ordinal) THEN
o  7149            search_xcb := #ADDRESS (1, ijl_p^.ajl_ordinal + mtc$job_fixed_segment,
o  7150              tmv$ptl_p^ [search_ptlo].xcb_offset);
o  7151            IF (search_xcb^.relative_task_priority > relative_priority) THEN
o  7152              insert_ptlo := search_ptlo;
o  7153            IFEND;
o  7154          IFEND;
o  7155          search_ptlo := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7156        WHILEND;
o  7157
o  7158        IF insert_ptlo <> 0 THEN
o  7159          IF ptlo = dcte.queue_head THEN
o  7160            dcte.queue_head := tmv$ptl_p^ [ptlo].ptl_thread;
o  7161          IFEND;
o  7162          IF ptlo = dcte.minor_priority THEN
o  7163            dcte.minor_priority := tmv$ptl_p^ [ptlo].ptl_thread;
o  7164          IFEND;
o  7165          IF insert_ptlo = dcte.major_priority THEN
o  7166            dcte.major_priority := ptlo;
o  7167          IFEND;
o  7168          tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [ptlo].ptl_thread;
o  7169          tmv$ptl_p^ [ptlo].ptl_thread := tmv$ptl_p^ [insert_ptlo].ptl_thread;
o  7170          tmv$ptl_p^ [insert_ptlo].ptl_thread := ptlo;
o  7171          IF tmv$ptl_p^ [ptlo].ptl_thread = 0 THEN
o  7172            dcte.queue_tail := ptlo;
o  7173          IFEND;
o  7174        IFEND;
o  7175      IFEND;
o  7176
o  7177    PROCEND relative_insert_minor_dct;
```

```
o  7179
o  7180   PROCEDURE  [INLINE] relative_insert_major_dct
o  7181     (     relative_priority: 0 .. 255;
o  7182           ptlo: ost$task_index;
o  7183           ijl_p: ^jmt$initiated_job_list_entry;
o  7184      VAR dcte: tmt$dct_entry);
o  7185
o  7186 { The following procedure is responsible for making sure a task which was
o  7187 { inserted in the MAJOR timeslice position is corrected prioritized with the
o  7188 { other tasks of the same job. This job is currently running with relative
o  7189 { prioritization enabled. The position of tasks of other jobs can only be
o  7190 { improved, they will never regress.
o  7191
o  7192      VAR
o  7193        insert_ptlo: ost$task_index,
o  7194        next_search_ptlo: ost$task_index,
o  7195        save_ptlo: ost$task_index,
o  7196        search_ptlo: ost$task_index,
o  7197        search_xcb: ^ost$execution_control_block;
o  7198
o  7199   { Insert_ptlo is the most recently moved ptlo.
o  7200   { Search_ptlo is the task which we are currently testing for
o  7201   { possible movement.
o  7202   { Save_ptlo is the last task which was NOT moved.
o  7203
o  7204      insert_ptlo := ptlo;
o  7205      search_ptlo := dcte.queue_head;
o  7206      save_ptlo := 0;
o  7207
o  7208   { The task has been positioned in the DCT. Now, all the tasks of that
o  7209   { job (currently in DCT) must be prioritized. The task has been placed
o  7210   { in the major priority position. Tasks above that point are checked to
o  7211   { determine if any of them have to be moved below the new task. If the
o  7212   { task is in the same job, and has a lower relative priority, the task is
o  7213   { moved below the new task.
o  7214
o  7215      WHILE (search_ptlo <> ptlo) DO
o  7216        next_search_ptlo := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7217        IF (tmv$ptl_p^ [search_ptlo].ijl_ordinal = tmv$ptl_p^ [ptlo].ijl_ordinal) THEN
o  7218          search_xcb := #ADDRESS (1, ijl_p^.ajl_ordinal + mtc$job_fixed_segment,
o  7219            tmv$ptl_p^ [search_ptlo].xcb_offset];
o  7220          IF (search_xcb^.relative_task_priority < relative_priority) THEN
o  7221            IF search_ptlo = dcte.queue_head THEN
o  7222              dcte.queue_head := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7223            ELSE
o  7224              tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7225            IFEND;
o  7226            IF search_ptlo = dcte.minor_priority THEN
o  7227              dcte.minor_priority := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7228            IFEND;
o  7229            tmv$ptl_p^ [search_ptlo].ptl_thread := tmv$ptl_p^ [insert_ptlo].ptl_thread;
o  7230            tmv$ptl_p^ [insert_ptlo].ptl_thread := search_ptlo;
o  7231            insert_ptlo := search_ptlo;
o  7232          ELSE
o  7233            save_ptlo := search_ptlo;
o  7234          IFEND;
```



```
o  7235        ELSE
o  7236          save_ptlo := search_ptlo;
o  7237        IFEND;
o  7238        search_ptlo := next_search_ptlo;
o  7239      WHILEND;
o  7240
o  7241      IF insert_ptlo <> ptlo THEN
o  7242        dcte.major_priority := insert_ptlo;
o  7243        IF tmv$ptl_p^ [insert_ptlo].ptl_thread = 0 THEN
o  7244          dcte.queue_tail := insert_ptlo;
o  7245        IFEND;
o  7246      ELSE
o  7247
o  7248 { There were not any tasks moved below the new task. We must now check to
o  7249 { see if there are any tasks below the new task, which have a relative priority
o  7250 { higher than the new task. A search is made of all of the tasks below the new
o  7251 { task, the new task is moved below the last task (of the same job) which
o  7252 { has a higher relative priority than the new task.
o  7253
o  7254        insert_ptlo := 0;
o  7255        search_ptlo := tmv$ptl_p^ [ptlo].ptl_thread;
o  7256        WHILE search_ptlo <> 0 DO
o  7257          IF (tmv$ptl_p^ [search_ptlo].ijl_ordinal = tmv$ptl_p^ [ptlo].ijl_ordinal) THEN
o  7258            search_xcb := #ADDRESS (1, ijl_p^.ajl_ordinal + mtc$job_fixed_segment,
o  7259              tmv$ptl_p^ [search_ptlo].xcb_offset];
o  7260            IF (search_xcb^.global_task_id <> tmv$null_global_task_id) AND
o  7261              (search_xcb^.relative_task_priority > relative_priority) THEN
o  7262              insert_ptlo := search_ptlo;
o  7263            IFEND;
o  7264          IFEND;
o  7265          search_ptlo := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7266        WHILEND;
o  7267        IF insert_ptlo <> 0 THEN
o  7268          IF ptlo = dcte.queue_head THEN
o  7269            dcte.queue_head := tmv$ptl_p^ [ptlo].ptl_thread;
o  7270          IFEND;
o  7271          IF ptlo = dcte.minor_priority THEN
o  7272            dcte.minor_priority := tmv$ptl_p^ [ptlo].ptl_thread;
o  7273          IFEND;
o  7274          IF ptlo = dcte.major_priority THEN
o  7275            dcte.major_priority := tmv$ptl_p^ [ptlo].ptl_thread;
o  7276          IFEND;
o  7277          tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [ptlo].ptl_thread;
o  7278          tmv$ptl_p^ [ptlo].ptl_thread := tmv$ptl_p^ [insert_ptlo].ptl_thread;
o  7279          tmv$ptl_p^ [insert_ptlo].ptl_thread := ptlo;
o  7280          IF tmv$ptl_p^ [ptlo].ptl_thread = 0 THEN
o  7281            dcte.queue_tail := ptlo;
o  7282          IFEND;
o  7283        IFEND;
o  7284      IFEND;
o  7285
o  7286    PROCEND relative_insert_major_dct;
```

RELATIVE_INSERT_TAIL_DCT

```
o  7288
o  7289    PROCEDURE  [INLINE]  relative_insert_tail_dct
o  7290      (     relative_priority: 0 .. 255;
o  7291            ptlo: ost$task_index;
o  7292            ijle_p: ^jmt$initiated_job_list_entry;
o  7293       VAR dcte: tmt$dct_entry);
o  7294
o  7295    { The new task has been placed at the tail of the DCT.
o  7296    { All of the tasks in the DCT are scanned. Any tasks (of the
o  7297    { same job) which have a lower relative priority than the new
o  7298    { task, are moved below the new task.
o  7299
o  7300      VAR
o  7301        insert_ptlo: ost$task_index,
o  7302        next_search_ptlo: ost$task_index,
o  7303        save_ptlo: ost$task_index,
o  7304        search_ptlo: ost$task_index,
o  7305        search_xcb: ^ost$execution_control_block;
o  7306
o  7307    { Insert_ptlo is the most recently moved ptlo.
o  7308    { Search_ptlo is the task which we are currently testing for
o  7309    { possible movement.
o  7310    { Save_ptlo is the last task which was NOT moved.
o  7311
o  7312      search_ptlo := dcte.queue_head;
o  7313      save_ptlo := 0;
o  7314      insert_ptlo := ptlo;
o  7315
o  7316
o  7317      WHILE (search_ptlo <> ptlo) DO
o  7318        next_search_ptlo := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7319        IF (tmv$ptl_p^ [search_ptlo].ijl_ordinal = tmv$ptl_p^ [ptlo].ijl_ordinal) THEN
o  7320          search_xcb := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment,
o  7321            tmv$ptl_p^ [search_ptlo].xcb_offset);
o  7322          IF (search_xcb^.relative_task_priority < relative_priority) THEN
o  7323            IF (search_ptlo = dcte.queue_head) THEN
o  7324              dcte.queue_head := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7325            IFEND;
o  7326            IF (search_ptlo = dcte.minor_priority) THEN
o  7327              dcte.minor_priority := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7328            IFEND;
o  7329            IF (search_ptlo = dcte.major_priority) THEN
o  7330              dcte.major_priority := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7331            IFEND;
o  7332            IF save_ptlo <> 0 THEN
o  7333              tmv$ptl_p^ [save_ptlo].ptl_thread := tmv$ptl_p^ [search_ptlo].ptl_thread;
o  7334            IFEND;
o  7335            tmv$ptl_p^ [search_ptlo].ptl_thread := 0;
o  7336            tmv$ptl_p^ [insert_ptlo].ptl_thread := search_ptlo;
o  7337            insert_ptlo := search_ptlo;
o  7338          ELSE
o  7339            save_ptlo := search_ptlo;
o  7340          IFEND;
o  7341        ELSE
o  7342          save_ptlo := search_ptlo;
o  7343        IFEND;
```


RELATIVE_INSERT_TAIL_DCT

```
o  7344        search_ptlo := next_search_ptlo;
o  7345      WHILEND;
o  7346
o  7347      dcte.queue_tail := insert_ptlo;
o  7348
o  7349    PROCEND relative_insert_tail_dct;
```

INSERT_IJL

```
o  7351
o  7352   PROCEDURE [INLINE] insert_ijl
o  7353     (     taskid: ost$global_task_id;
o  7354           cst_p: ^ost$cpu_state_table);
o  7355
o  7356 {
o  7357 {  The purpose of this procedure is to insert the selected
o  7358 {  task's PTL entry in the IJL thread for it's job.  The
o  7359 {  first task in the IJL thread is the job monitor task.
o  7360 {
o  7361 {    INSERT_IJL (TASKID)
o  7362 {
o  7363 {  TASKID : (INPUT) This parameter specifies the taskid of
o  7364 {                   of the specified task.
o  7365 {
o  7366 {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
o  7367 {
o  7368
o  7369     tmv$ptl_p^ [taskid.index].ijl_thread := tmv$ptl_p^ [cst_p^.ijle_p^.job_monitor_taskid.index].ijl_thread;
o  7370     tmv$ptl_p^ [cst_p^.ijle_p^.job_monitor_taskid.index].ijl_thread := taskid.index;
o  7371
o  7372   PROCEND insert_ijl;
```


REMOVE_IJL

```
o  7374
o  7375   PROCEDURE [INLINE] remove_ijl
o  7376     (     taskid: ost$global_task_id;
o  7377           cst_p: ^ost$cpu_state_table);
o  7378
o  7379
o  7380 {
o  7381 {  The purpose of this procedure is to remove the PTL
o  7382 {  entry for the specified task from the IJL thread.
o  7383 {
o  7384 {    REMOVE_IJL (TASKID)
o  7385 {
o  7386 {  TASKID : (INPUT) This parameter specifies the taskid
o  7387 {                   of the selected task.
o  7388 {
o  7389 {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
o  7390 {
o  7391
o  7392     VAR
o  7393       last_ptlo,
o  7394       curr_ptlo: ost$task_index;
o  7395
o  7396
o  7397
o  7398     last_ptlo := 0;
o  7399     curr_ptlo := cst_p^.ijle_p^.job_monitor_taskid.index;
o  7400
o  7401   /search_loop/
o  7402     WHILE curr_ptlo <> 0 DO
o  7403       IF curr_ptlo = taskid.index THEN
o  7404         EXIT /search_loop/;
o  7405       IFEND;
o  7406       last_ptlo := curr_ptlo;
o  7407       curr_ptlo := tmv$ptl_p^ [last_ptlo].ijl_thread;
o  7408     WHILEND /search_loop/;
o  7409
o  7410     tmv$ptl_p^ [last_ptlo].ijl_thread := tmv$ptl_p^ [curr_ptlo].ijl_thread;
o  7411     tmv$ptl_p^ [curr_ptlo].ijl_thread := 0;
o  7412
o  7413   PROCEND remove_ijl;
```

TMP$FIND_XCB

```
        0  7415
        0  7416    PROCEDURE [XDCL] tmp$find_xcb
        0  7417      (     taskid: ost$global_task_id;
        0  7418       VAR xcb_p: ^ost$execution_control_block;
        0  7419       VAR ijle_p: ^jmt$initiated_job_list_entry;
        0  7420       VAR status: syt$monitor_status);
        0  7421
        0  7422
        0  7423 {
        0  7424 {  The purpose of this request is to get the pointer to the execution control
        0  7425 {  block of a specified task.  If the job is swapped but ID has not been initiated
        0  7426 {  an ajl entry is assigned to allow access to the xcb.  Otherwise an
        0  7427 {  error is returned.
        0  7428 {
        0  7429 {    TMP$FIND_XCB (TASKID, XCB_P, IJLE_P, STATUS)
        0  7430 {
        0  7431 {  TASKID : (INPUT) This parameter specifies the taskid of the
        0  7432 {                   selected task.
        0  7433 {
        0  7434 {  XCB_P : (OUTPUT) This parameter specifies the pointer to the
        0  7435 {                   execution control block.
        0  7436 {
        0  7437 {  IJLE_P : (OUTPUT) This parameter specifies the pointer to the
        0  7438 {                   initiated job list entry.
        0  7439 {
        0  7440 {  STATUS : (OUTPUT) This parameter specifies the request status.
        0  7441 {
        0  7442 {  NOTE : This procedure will lock the ajl entry of the job if xcb access is
        0  7443 {         possible.  It is the callers responsibility to unlock the ajl with
        0  7444 {         a call to jmp$free_ajl_entry.
        0  7445
        0  7446     VAR
        0  7447       inhibit_access: boolean;
        0  7448
        0  7449     status.normal := TRUE;
        4  7450     tmp$set_lock (tmv$ptl_lock);
       4A  7451
       4A  7452     tmp$check_taskid_with_lock_set (taskid, tmc$opt_return, status);
       AC  7453     IF NOT status.normal THEN
       B4  7454       tmp$clear_lock (tmv$ptl_lock);
       E6  7455       RETURN;
       E8  7456     IFEND;
       E8  7457
       E8  7458     jmp$get_ijle_p (tmv$ptl_p^ [taskid.index].ijl_ordinal, ijle_p);
       E8  7459     tmp$get_xcb_access_status (ijle_p, tmv$ptl_p^ [taskid.index].ijl_ordinal, inhibit_access);
      1A2  7460     IF inhibit_access THEN
      1AA  7461       mtp$set_status_abnormal ('TM', tme$job_swapped_out, status);
      1BE  7462     ELSE
      1BE  7463       xcb_p := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [taskid.index].
      1F0  7464             xcb_offset);
      1F0  7465     IFEND;
      1F0  7466     tmp$clear_lock (tmv$ptl_lock);
      222  7467
      222  7468   PROCEND tmp$find_xcb;
        0  7469
```


TMP$GET_XCB_P

```
        0  7471
        0  7472   PROCEDURE [XDCL] tmp$get_xcb_p
        0  7473     (     taskid: ost$global_task_id;
        0  7474      VAR xcb_p: ^ost$execution_control_block;
        0  7475      VAR ijle_p: ^jmt$initiated_job_list_entry);
        0  7476
        0  7477 {
        0  7478 {   The purpose of this request is to get a pointer to the XCB of the task
        0  7479 { specified by a global task id (GTID). If the GTID is invalid, the system is halted
        0  7480 { with an 'taskid error'. If the taskid is valid but the task is swapped out,
        0  7481 { a NIL pointer is returned.  A pointer to the initiated job list entry is also
        0  7482 { returned.
        0  7483 {
        0  7484 {         tmp$get_xcb_p (taskid, xcb_p, ijle_p);
        0  7485 {
        0  7486 { TASKID: (input) This parameter specifies the GTID of the task to be located.
        0  7487 {
        0  7488 { XCB_P: (output) This parameter contains a pointer to the XCB of the specified
        0  7489 {          task. If the task is swapped out, a NIL pointer is returned.
        0  7490 {
        0  7491 { IJLE_P: (output) This parameter contains a pointer to the initiated job list
        0  7492 {          entry of the job.
        0  7493
        0  7494 {  NOTE: This procedure will increment the ajl in use count or assign an ajl if
        0  7495 {         xcb access is possible.  It is the callers responsibility to decrement it
        0  7496 {         with a call to jmp$free_ajl_entry.
        0  7497
        0  7498     VAR
        0  7499       inhibit_access: boolean;
        0  7500
        0  7501     tmp$set_lock (tmv$ptl_lock);
       42  7502     tmp$stop_if_bad_taskid (taskid);
       A8  7503     jmp$get_ijle_p (tmv$ptl_p^ [taskid.index].ijl_ordinal, ijle_p);
       A8  7504
       A8  7505     tmp$get_xcb_access_status (ijle_p, tmv$ptl_p^ [taskid.index].ijl_ordinal, inhibit_access);
      164  7506     IF inhibit_access THEN
      16C  7507       xcb_p := NIL;
      17C  7508     ELSE
      17C  7509       xcb_p := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [taskid.index].
      1AE  7510             xcb_offset);
      1AE  7511     IFEND;
      1AE  7512     tmp$clear_lock (tmv$ptl_lock);
      1E2  7513
      1E2  7514   PROCEND tmp$get_xcb_p;
```

TMP$TEST_GET_XCB_P

```
        0  7516
        0  7517    PROCEDURE [XDCL] tmp$test_get_xcb_p
        0  7518      (     taskid: ost$global_task_id;
        0  7519       VAR xcb_p: ^ost$execution_control_block;
        0  7520       VAR ijle_p: ^jmt$initiated_job_list_entry);
        0  7521
        0  7522 {  NOTE: This procedure will increment the ajl in use count or assign an ajl if
        0  7523 {        xcb access is possible.  It is the callers responsibility to decrement it
        0  7524 {        with a call to jmp$free_ajl_entry.
        0  7525
        0  7526      VAR
        0  7527        inhibit_access: boolean;
        0  7528
        0  7529      tmp$set_lock (tmv$ptl_lock);
       42  7530
       42  7531      IF (taskid = tmv$null_global_task_id) OR (taskid.index > UPPERBOUND (tmv$ptl_p^)) OR
       84  7532            (tmv$ptl_p^ [taskid.index].sequence_number <> taskid.seqno) OR
       84  7533            (tmv$ptl_p^ [taskid.index].status = tmc$ts_null) THEN
       84  7534        xcb_p := NIL;
       84  7535        ijle_p := NIL;
       9C  7536      ELSE
       9C  7537        jmp$get_ijle_p (tmv$ptl_p^ [taskid.index].ijl_ordinal, ijle_p);
       9C  7538
       9C  7539        tmp$get_xcb_access_status (ijle_p, tmv$ptl_p^ [taskid.index].ijl_ordinal, inhibit_access);
      14C  7540        IF inhibit_access THEN
      154  7541          xcb_p := NIL;
      164  7542        ELSE
      164  7543          xcb_p := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [taskid.index].
      196  7544                xcb_offset);
      196  7545        IFEND;
      196  7546      IFEND;
      196  7547      tmp$clear_lock (tmv$ptl_lock);
      1CA  7548
      1CA  7549    PROCEND tmp$test_get_xcb_p;
```


TMP$GET_TOP_OF_STACK

```
        0  7551 {
        0  7552 { PURPOSE:
        0  7553 {   This procedure is used to fetch the value of the TOP-OF-STACK pointer for a specified
        0  7554 {   ring of a specified task.
        0  7555 { DESIGN:
        0  7556 {   The value of the TOS register is determined from the exchange package as follows:
        0  7557 {   If the value of TOS cannot be determined because the task is executing on another
        0  7558 {   processor of a dual CPU system, TOS cannot be determined; a value of 2**31-1 is
        0  7559 {   returned.
        0  7560 {
        0  7561
        0  7562    PROCEDURE [XDCL] tmp$get_top_of_stack
        0  7563      (     taskid: ost$global_task_id;
        0  7564            ring: ost$valid_ring;
        0  7565       VAR tos: integer);
        0  7566
        0  7567      VAR
        0  7568        cst_p: ^ost$cpu_state_table,
        0  7569        ijle_p: ^jmt$initiated_job_list_entry,
        0  7570        inhibit_access: boolean,
        0  7571        xcb_p: ^ost$execution_control_block;
        0  7572
        0  7573      tmp$set_lock (tmv$ptl_lock);
       46  7574      tmp$stop_if_bad_taskid (taskid);
       AC  7575      jmp$get_ijle_p (tmv$ptl_p^ [taskid.index].ijl_ordinal, ijle_p);
       AC  7576
       AC  7577      tmp$get_xcb_access_status (ijle_p, tmv$ptl_p^ [taskid.index].ijl_ordinal, inhibit_access);
      162  7578      IF inhibit_access THEN
      16A  7579        mtp$error_stop ('TM - XCB not accessible');
      18E  7580      ELSE
      18E  7581        xcb_p := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [taskid.index].
      1BA  7582              xcb_offset);
      1BA  7583      IFEND;
      1BA  7584
      1BA  7585      mtp$cst_p (cst_p);
      1CA  7586
      1CA  7587      IF ((tmv$ptl_p^ [taskid.index].status = tmc$ts_executing) AND (cst_p^.taskid <> taskid)) OR
      20E  7588         (xcb_p^.stack_pages_saved [ring] = TRUE) THEN
      20E  7589        tos := 7fffffff(16);                    { TOS cannot be determined if executing on another processor.}
      21E  7590      ELSEIF xcb_p^.xp.p_register.pva.ring > ring THEN
      22C  7591        tos := 0;                               { TOS is zero if executing in a higher ring.}
      23A  7592      ELSEIF xcb_p^.xp.p_register.pva.ring = ring THEN
      23E  7593        tos := #OFFSET (xcb_p^.xp.a0_dynamic_space_pointer); {TOS in XP.A0  is correct if in same ring.}
      252  7594      ELSE
      252  7595        tos := xcb_p^.xp.tos_registers [ring].pva.offset; {TOS in XP.TOS is correct if in lower ring.}
      268  7596      IFEND;
      268  7597
      268  7598      jmp$unlock_ajl_with_lock (ijle_p);
      2AA  7599      tmp$clear_lock (tmv$ptl_lock);
      2DE  7600
      2DE  7601    PROCEND tmp$get_top_of_stack;
        0  7602
```

TMP$GET_XCB_P_FROM_PTLO

```
        o  7604
        o  7605 {  NOTE : Any procedure calling this routine must have tmv$ptl_lock set.
        o  7606
        o  7607    PROCEDURE [INLINE] tmp$get_xcb_p_from_ptlo
        o  7608      (     ptlo: ost$task_index;
        o  7609            ajl_ordinal: jmt$ajl_ordinal;
        o  7610       VAR xcb_p: ^ost$execution_control_block);
        o  7611
        o  7612      xcb_p := #ADDRESS (1, ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [ptlo].xcb_offset);
        o  7613
        o  7614    PROCEND tmp$get_xcb_p_from_ptlo;
        o  7615
```


TMP$FIND_NEXT_XCB

```
        o  7617
        o  7618    PROCEDURE [XDCL] tmp$find_next_xcb
        o  7619      (     search: tmt$fnx_search_type;
        o  7620            ijle_p: ^jmt$initiated_job_list_entry;
        o  7621            ijl_ordinal: jmt$ijl_ordinal;
        o  7622       VAR state: tmt$find_next_xcb_state;
        o  7623       VAR xcb_p: ^ost$execution_control_block);
        o  7624
        o  7625 {
        o  7626 {  The purpose of this request is to scan the PTL entry for each
        o  7627 {  task in a JOB or the SYSTEM and return its XCB_P.
        o  7628 {
        o  7629 {  SEARCH: (INPUT) This parameter specifies the type of search to be
        o  7630 {                   done, JOB, SYSTEM, SWAPPING_JOB, of CONTINUE
        o  7631 {                   previous search.
        o  7632 {
        o  7633 {  IJLE_P: (INPUT) This parameter specifies a pointer to the initiated
        o  7634 {                   job list entry of the job to be searched.
        o  7635 {
        o  7636 {  IJL_ORDINAL: (INPUT) This parameter specifies the initiated job list
        o  7637 {                       ordinal of the job to be searched.
        o  7638 {
        o  7639 {  STATE: (INPUT,OUTPUT) This parameter specifies the type of search
        o  7640 {                        that had been started by a previous call.
        o  7641 {
        o  7642 {  XCB_P: (OUTPUT) This parameter specifies the XCB_P of the next task
        o  7643 {                   in the JOB or SYSTEM.
        o  7644 {
        o  7645
        o  7646      VAR
        o  7647        ijl_bi: jmt$ijl_block_index,
        o  7648        ijl_bn: jmt$ijl_block_number,
        o  7649        inhibit_access: boolean,
        o  7650        next_ijle_p: ^jmt$initiated_job_list_entry,
        o  7651        start_index: integer;
        o  7652
        o  7653     tmp$set_lock (tmv$ptl_lock);
       42  7654
       42  7655 {Initialize the search state variable if this is not a continuation search.
       42  7656
       42  7657     IF search <> tmc$fnx_continue THEN
       4A  7658       state.search := search;
       4A  7659       IF search = tmc$fnx_system THEN
       58  7660         state.ajl_ordinal := jmv$system_ajl_ordinal;
       58  7661         state.ijl_ordinal := jmv$system_ijl_ordinal;
       58  7662         state.in_use_incremented := TRUE;
       58  7663         jmv$ajl_p^ [jmv$system_ajl_ordinal].in_use := jmv$ajl_p^ [jmv$system_ajl_ordinal].in_use +
       58  7664               jmc$lock_ajl;
       58  7665         jmp$get_ijle_p (jmv$system_ijl_ordinal, state.ijle_p);
       58  7666         state.next_ptlo := jmv$ajl_p^ [state.ajl_ordinal].ijle_p^.job_monitor_taskid.index;
       D2  7667       ELSE
       D2  7668         tmp$get_xcb_access_status (ijle_p, ijl_ordinal, inhibit_access);
      164  7669         state.in_use_incremented := NOT inhibit_access;
      164  7670         IF (search = tmc$fnx_job) AND inhibit_access THEN
      17C  7671           state.next_ptlo := 0;
      186  7672         ELSE
```

TMP$FIND_NEXT_XCB

```
186  7673            state.ajl_ordinal := ijle_p^.ajl_ordinal;
186  7674            state.next_ptlo := ijle_p^.job_monitor_taskid.index;
186  7675            state.ijl_ordinal := ijl_ordinal;
186  7676            state.ijle_p := ijle_p;
1AC  7677          IFEND;
1AC  7678        IFEND;
1BO  7679
1BO  7680 { If a full system search is in process and we have reached the end of the current job.
1BO  7681 { Release the temporary ajl ordinal if necessary.  Find the next job to be searched.
1BO  7682
1BO  7683      ELSEIF (state.next_ptlo = 0) AND (state.search = tmc$fnx_system) THEN
1C6  7684
1C6  7685        jmp$unlock_ajl_with_lock (state.ijle_p);
20C  7686        state.in_use_incremented := FALSE;
20C  7687
20C  7688        start_index := state.ijl_ordinal.block_index + 1;
20C  7689
20C  7690      /find_next_job/
20C  7691        FOR ijl_bn := state.ijl_ordinal.block_number TO jmv$ijl_p.max_block_in_use DO
232  7692          IF jmv$ijl_p.block_p^ [ijl_bn].index_p <> NIL THEN
244  7693            FOR ijl_bi := start_index TO UPPERVALUE (jmt$ijl_block_index) DO
24E  7694              next_ijle_p := ^jmv$ijl_p.block_p^ [ijl_bn].index_p^ [ijl_bi];
24E  7695              IF next_ijle_p^.entry_status <> jmc$ies_entry_free THEN
268  7696                state.ijl_ordinal.block_index := ijl_bi;
268  7697                state.ijl_ordinal.block_number := ijl_bn;
268  7698                tmp$get_xcb_access_status (next_ijle_p, state.ijl_ordinal, inhibit_access);
306  7699                state.in_use_incremented := NOT inhibit_access;
306  7700                IF NOT inhibit_access THEN
316  7701                  state.next_ptlo := next_ijle_p^.job_monitor_taskid.index;
316  7702                  state.ajl_ordinal := next_ijle_p^.ajl_ordinal;
316  7703                  state.ijle_p := next_ijle_p;
316  7704                  EXIT /find_next_job/;
32E  7705                IFEND;
32E  7706              IFEND;
32E  7707            FOREND;
332  7708            start_index := LOWERVALUE (jmt$ijl_block_index);
336  7709          IFEND;
336  7710        FOREND /find_next_job/;
33A  7711
33A  7712      IFEND;
33A  7713
33A  7714      IF state.next_ptlo = 0 THEN
342  7715        IF state.in_use_incremented THEN
34A  7716          jmp$unlock_ajl_with_lock (state.ijle_p);
390  7717          state.in_use_incremented := FALSE;
396  7718        IFEND;
396  7719        xcb_p := NIL;
3A6  7720      ELSE
3A6  7721        xcb_p := #ADDRESS (1, mtc$job_fixed_segment + state.ajl_ordinal, tmv$ptl_p^ [state.next_ptlo].
3A6  7722              xcb_offset);
3A6  7723        state.next_ptlo := tmv$ptl_p^ [state.next_ptlo].ijl_thread;
3E0  7724      IFEND;
3E0  7725      tmp$clear_lock (tmv$ptl_lock);
41C  7726
41C  7727    PROCEND tmp$find_next_xcb;
```


TMP$SET_TASK_WAIT

```
  0  7729
  0  7730    PROCEDURE [XDCL] tmp$set_task_wait
  0  7731      (     task_status: tmt$task_status);
  0  7732
  0  7733
  0  7734 {  The purpose of this procedure is to change the status
  0  7735 {  of the current task to a specified WAIT status.
  0  7736 {
  0  7737 {     TMP$SET_TASK_WAIT (TASK_STATUS)
  0  7738 {
  0  7739 {  TASK_STATUS : (INPUT) This parameter specifies the new task
  0  7740 {                         status.
  0  7741 {
  0  7742
  0  7743      VAR
  0  7744        cst_p: ^ost$cpu_state_table,
  0  7745        taskid: ost$global_task_id;
  0  7746
  0  7747      mtp$cst_p (cst_p);
 14  7748      tmp$set_lock (tmv$ptl_lock);
 60  7749      IF (task_status >= tmc$ts_first_external_queue) OR (tmv$ptl_p^ [cst_p^.taskid.index].new_task_status >
 86  7750            tmc$ts_null) THEN
 86  7751        mtp$error_stop ('TM37 - bad call to set_task_wait');
 A6  7752      IFEND;
 A6  7753      IF (task_status = tmc$ts_io_wait_not_queued) THEN
 B0  7754        tmv$io_wait_task_count := tmv$io_wait_task_count + 1;
 BE  7755      IFEND;
 BE  7756      tmv$ptl_p^ [cst_p^.taskid.index].new_task_status := task_status;
 BE  7757      tmp$clear_lock (tmv$ptl_lock);
110  7758      cst_p^.dispatch_control.call_dispatcher := TRUE;
110  7759      IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
120  7760        cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
128  7761      IFEND;
128  7762
128  7763    PROCEND tmp$set_task_wait;
```

```
        0  7765
        0  7766    PROCEDURE [XDCL] tmp$set_task_ready
        0  7767      (     taskid: ost$global_task_id;
        0  7768            readying_task_priority: jmt$dispatching_priority;
        0  7769            ready_condition: tmt$ready_condition);
        0  7770
        0  7771
        0  7772 {
        0  7773 {  The purpose of this procedure is to make a task ready either
        0  7774 {  conditionally or unconditionally.  A task can be conditionally
        0  7775 {  made ready if it is currently in a wait status and can be
        0  7776 {  unconditionally made ready irrespective of it's current status.
        0  7777 {  The WAIT INHIBITED flag in the PTL is set if it is specified
        0  7778 {  on the request.
        0  7779 {
        0  7780 {     TMP$SET_TASK_READY (TASKID, READY_CONDITION)
        0  7781 {
        0  7782 {  TASKID : (INPUT) This parameter specifies the task to be made ready.
        0  7783 {
        0  7784 {  READYING_TASK_PRIORITY : (INPUT) This parameter specifies the
        0  7785 {                           priority of the task which has issued the
        0  7786 {                           request to ready this task. This value is only used
        0  7787 {                           if the task being readied has subsystem locks set.
        0  7788 {
        0  7789 {  READY_CONDITION : (INPUT) This parameter specifies if the task
        0  7790 {                             is to be made ready conditionally or
        0  7791 {                             unconditionally.
        0  7792 {
        0  7793
        0  7794      VAR
        0  7795        attempt_preselection: boolean,
        0  7796        cst_p: ^ost$cpu_state_table,
        0  7797        old_dispatching_priority: jmt$dispatching_priority,
        0  7798        ijle_p: ^jmt$initiated_job_list_entry,
        0  7799        null_dispatching_info: jmt$dispatching_control_info,
        0  7800        psuedo_rb: tmt$rb_update_job_task_enviro,
        0  7801        ptl_p: ^tmt$primary_task_list_entry,
        0  7802        xcb_p: ^ost$execution_control_block,
        0  7803        task_priority_changed: boolean;
        0  7804
        0  7805
        0  7806      tmp$set_lock (tmv$ptl_lock);
       42  7807      tmp$stop_if_bad_taskid (taskid);
       A8  7808      ptl_p := ^tmv$ptl_p^ [taskid.index];
       A8  7809      jmp$get_ijle_p (ptl_p^.ijl_ordinal, ijle_p);
       A8  7810      xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, ptl_p^.xcb_offset);
       A8  7811
       A8  7812      IF (ijle_p^.interactive_task_gtid = taskid) THEN
       FE  7813        psuedo_rb.reqcode := syc$rc_update_job_task_enviro;
       FE  7814        psuedo_rb.subcode := tmc$ujte_dispatching_priority;
       FE  7815        psuedo_rb.request_origin := tmc$cpo_interactive_command;
       FE  7816        psuedo_rb.ijl_ordinal := ptl_p^.ijl_ordinal;
       FE  7817        psuedo_rb.system_supplied_name := ijle_p^.system_supplied_name;
      11E  7818        psuedo_rb.dispatching_control_info := null_dispatching_info;
      11E  7819        old_dispatching_priority := ijle_p^.dispatching_control.dispatching_priority;
      11E  7820        mtp$cst_p (cst_p);
```



```
      13A  7821        tmp$mtr_update_job_task_enviro (psuedo_rb, cst_p);
      164  7822        IF (ijle_p^.dispatching_control.dispatching_priority <> old_dispatching_priority) AND
      17A  7823          (ijle_p^.entry_status = jmc$ies_swapin_candidate) THEN
      17A  7824          jmv$subsystem_priority_changes [ijle_p^.job_scheduler_data.service_class] := TRUE;
      17A  7825          jmp$set_scheduler_event (jmc$subsystem_priority_change);
      198  7826        IFEND;
      198  7827        ijle_p^.interactive_task_gtid := tmv$null_global_task_id;
      1A4  7828      IFEND;
      1A4  7829
      1A4  7830 { Tasks with subsystem locks set will execute with the higher
      1A4  7831 {  of the task's or the "readying" task's priority, after it
      1A4  7832 { has run for the subsystem lock threshold with a priority of P9.
      1A4  7833
      1A4  7834      task_priority_changed := FALSE;
      1A4  7835
      1A4  7836      IF (readying_task_priority <> 0) AND ((ptl_p^.ptl_flags.subsystem_locks_set) OR
      1CE  7837        ((ptl_p^.idle_status < tmc$is_idled) AND (xcb_p^.system_table_lock_count > 0)))  THEN
      1CE  7838        IF readying_task_priority > ptl_p^.readying_task_priority THEN
      1DA  7839          ptl_p^.readying_task_priority := readying_task_priority;
      1DA  7840          task_priority_changed := TRUE;
      1E2  7841        IFEND;
      1E2  7842        IF ijle_p^.scheduling_dispatching_priority < readying_task_priority THEN
      1F2  7843          ijle_p^.scheduling_dispatching_priority := readying_task_priority;
      1F2  7844          jmp$subsystem_priority_change (ijle_p);
      208  7845        IFEND;
      20C  7846      IFEND;
      20C  7847
      20C  7848      IF (ptl_p^.status = tmc$ts_executing) OR (ptl_p^.status = tmc$ts_ready_and_selected) THEN
      21C  7849        IF ptl_p^.new_task_status < tmc$ts_first_ready_uncond THEN
      226  7850          ptl_p^.new_task_status := tmc$ts_null;
      22A  7851        IFEND;
      22E  7852
      22E  7853      ELSEIF ptl_p^.status = tmc$ts_ready THEN
      232  7854
      232  7855 { If the task being readied has subsystem locks set, it must be
      232  7856 { removed from the ready string and reinserted. It is possible
      232  7857 { that the task will execute at a different priority depending
      232  7858 { on how long the task has had subsystem locks set, and
      232  7859 { the priority of the readying task.
      232  7860
      232  7861        IF task_priority_changed AND (xcb_p^.system_table_lock_count > 0) AND
      254  7862             (xcb_p^.system_table_lock_count < osc$system_table_lock_set) AND
      254  7863          (xcb_p^.subsystem_lock_priority_count >= tmv$subsystem_prior_threshold) THEN
      254  7864          tmp$remove_task_from_dct (taskid.index);
      3C8  7865          attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*     7866          tmp$dct_ready_task (xcb_p, ijle_p, taskid.index, attempt_preselection);
      D0C  7867        IFEND;
      D10  7868
      D10  7869      ELSEIF ptl_p^.status < tmc$ts_ready_but_swapped THEN
      D16  7870
      D16  7871 { If the task is in the timed wiat queue, take it out.  Adjust the tasks_not_in_long_wait count based upon
      D16  7872 { the task status.  If the job the task belongs to is swapped, call scheduler to swap the job in.  If the
      D16  7873 { job is in memory, insert the task into the DCT.
      D16  7874
      D16  7875
      D16  7876        IF ptl_p^.status <= tmc$ts_last_status_in_wait_q THEN
```

TMP$SET_TASK_READY

```
      D1C  7877          tmp$remove_task_from_q (ptl_p, tmv$timed_wait_queue);
      D66  7878          IF (ptl_p^.status = tmc$ts_timeout_reqexp_longvlong) THEN
      D78  7879            ijle_p^.statistics.tasks_not_in_long_wait := ijle_p^.statistics.tasks_not_in_long_wait + 1;
      D86  7880          IFEND;
      D8A  7881        ELSE
      D8A  7882          IF (ptl_p^.status = tmc$ts_timeout_reqexp_infvlong) OR (ptl_p^.status =
      D96  7883                tmc$ts_timed_wait_not_queued) THEN
      D96  7884            ijle_p^.statistics.tasks_not_in_long_wait := ijle_p^.statistics.tasks_not_in_long_wait + 1;
      DA4  7885          IFEND;
      DA4  7886        IFEND;
      DA4  7887
      DA4  7888        IF ptl_p^.idle_status = tmc$is_idled THEN
      DAE  7889          ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      DAE  7890          ptl_p^.status := tmc$ts_ready_but_swapped;
      DAE  7891          ptl_p^.idle_status := tmc$is_idled_sched_notified;
      DAE  7892          jmp$ready_task_in_swapped_job (ptl_p^.ijl_ordinal, ijle_p);
      DE4  7893        ELSEIF ptl_p^.idle_status < tmc$is_idled THEN
      DE8  7894          ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      DE8  7895          ptl_p^.status := tmc$ts_ready;
      DE8  7896          jmv$ajl_p^ [ijle_p^.ajl_ordinal].job_is_good_swap_candidate := FALSE;
      DE8  7897          attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*     7898          tmp$dct_ready_task (xcb_p, ijle_p, taskid.index, attempt_preselection);
     170A  7899        IFEND;
     170A  7900
     170A  7901        ?IF NOT debug THEN
     170A  7902          IF mlv$c170_rqst_blk.req <> NIL THEN
     171C  7903            IF taskid = mlv$c170_rqst_blk.req^.task_id THEN
     1728  7904              mtv$mli_status.ready := TRUE;
     1730  7905            IFEND;
     1730  7906          IFEND;
     1734  7907        ?IFEND
     1734  7908
     1734  7909      ELSEIF ptl_p^.status = tmc$ts_ready_but_swapped THEN
     1738  7910        jmp$ready_task_in_swapped_job (ptl_p^.ijl_ordinal, ijle_p);
     1754  7911      IFEND;
     1754  7912
     1754  7913      IF ptl_p^.status = tmc$ts_segment_lock_wait THEN
     175E  7914        remove_task_from_seg_lock_q (taskid, ijle_p, ptl_p);
     1774  7915      IFEND;
     1774  7916
     1774  7917 {The 'wait_inhibited' flag in the PTL is set to tmc$wi_wait_selected only after
     1774  7918 {the successful completion of a wait request.  This prevents potential timing
     1774  7919 {conflicts between ready_task and wait requests.
     1774  7920
     1774  7921      IF (ready_condition = tmc$rc_ready_conditional_wi) AND
     1790  7922        (NOT (ptl_p^.ptl_flags.wait_inhibited = tmc$wi_wait_selected) AND
     1790  7923          NOT (ptl_p^.ptl_flags.wait_inhibited = tmc$wi_wait_selected_r3)) THEN
     1790  7924        ptl_p^.ptl_flags.wait_inhibited := tmc$wi_wait_inhibited;
     1798  7925      IFEND;
     1798  7926      ijle_p^.long_wait_aging_complete := FALSE;
     1798  7927      tmp$clear_lock (tmv$ptl_lock);
     17D6  7928
     17D6  7929    PROCEND tmp$set_task_ready;
```


TMP$SET_TASK_READY_UNCOND

```
        0  7931
        0  7932    PROCEDURE [XDCL] tmp$set_task_ready_uncond
        0  7933      (     taskid: ost$global_task_id;
        0  7934            task_status: tmt$task_status);
        0  7935
        0  7936
        0  7937 {  The purpose of this procedure is to unconditionally ready the
        0  7938 {  specified task. The task to be readied must be in the task
        0  7939 {  status specified on this request.
        0  7940 {
        0  7941 {    TMP$READY_TASK_UNCONDITIONAL (TASKID, TASKSTATUS)
        0  7942 {
        0  7943 {  TASKID : (OUTPUT) This parameter specifies the taskid of the
        0  7944 {                task to be readied.
        0  7945 {
        0  7946 {  TASKSTATUS: (INPUT) This parameter specifies the current status
        0  7947 {       of the task.
        0  7948
        0  7949      VAR
        0  7950        attempt_preselection: boolean,
        0  7951        ijle_p: ^jmt$initiated_job_list_entry,
        0  7952        xcb_p: ^ost$execution_control_block,
        0  7953        ptl_p: ^tmt$primary_task_list_entry;
        0  7954
        0  7955
        0  7956      tmp$set_lock (tmv$ptl_lock);
       42  7957      tmp$stop_if_bad_taskid (taskid);
       A8  7958      ptl_p := ^tmv$ptl_p^ [taskid.index];
       A8  7959      jmp$get_ijle_p (ptl_p^.ijl_ordinal, ijle_p);
       A8  7960      IF (task_status = tmc$ts_io_wait_not_queued) THEN
       E2  7961        tmv$io_wait_task_count := tmv$io_wait_task_count - 1;
       EC  7962      IFEND;
       EC  7963      IF ptl_p^.status <> task_status THEN
       F8  7964        IF (ptl_p^.status <> tmc$ts_executing) OR (ptl_p^.new_task_status <> task_status) THEN
      106  7965          mtp$error_stop ('TM96 - task not queued');
      126  7966        IFEND;
      126  7967        ptl_p^.new_task_status := tmc$ts_null;
      130  7968      ELSE
      130  7969        IF ptl_p^.idle_status = tmc$is_idled THEN
      13C  7970          ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      13C  7971          ptl_p^.status := tmc$ts_ready_but_swapped;
      13C  7972          jmp$ready_task_in_swapped_job (ptl_p^.ijl_ordinal, ijle_p);
      168  7973          ptl_p^.idle_status := tmc$is_idled_sched_notified;
      172  7974        ELSEIF ptl_p^.idle_status < tmc$is_idled THEN
      176  7975          ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      176  7976          ptl_p^.status := tmc$ts_ready;
      176  7977          attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
      19C  7978          xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, ptl_p^.xcb_offset);
      19C  7979          tmp$dct_ready_task (xcb_p, ijle_p, taskid.index, attempt_preselection);
      A76  7980        IFEND;
      A78  7981      IFEND;
      A78  7982
      A78  7983      tmp$clear_lock (tmv$ptl_lock);
      AAA  7984
      AAA  7985    PROCEND tmp$set_task_ready_uncond;
```

```
        o  7987
        o  7988    PROCEDURE [XDCL] tmp$reissue_monitor_request;
        o  7989
        o  7990  {
        o  7991  {   The purpose of this request is to cause the P register of the current task to
        o  7992  { be decremented by 2. This wil cause the monitor request issued by the task
        o  7993  { to be reissued when the task is next executed. This request does NOT change
        o  7994  { the status of the task or cause a task-switch. This request is intended
        o  7995  { to be used when the resources required to process the current request from the
        o  7996  { task are unavailable.
        o  7997  {
        o  7998  {         tmp$reissue_monitor_request;
        o  7999  {
        o  8000
        o  8001      VAR
        o  8002        cst_p: ^ost$cpu_state_table;
        o  8003
        o  8004      mtp$cst_p (cst_p);
       14  8005
       14  8006      cst_p^.xcb_p^.xp.p_register.pva.offset := cst_p^.xcb_p^.xp.p_register.pva.offset - 2;
       14  8007
       14  8008    PROCEND tmp$reissue_monitor_request;


        o  8010
        o  8011    PROCEDURE [XDCL] tmp$cause_task_switch;
        o  8012
        o  8013      VAR
        o  8014        cst_p: ^ost$cpu_state_table;
        o  8015
        o  8016      mtp$cst_p (cst_p);
       14  8017      tmp$set_lock (tmv$ptl_lock);
       60  8018      IF (tmv$ptl_p^ [cst_p^.taskid.index].new_task_status > tmc$ts_null) AND
       82  8019        (tmv$ptl_p^ [cst_p^.taskid.index].new_task_status <> tmc$ts_timeout_reqexp_shortshrt) THEN
       82  8020        mtp$error_stop ('TM38 - bad call to cause_task_switch');
       A2  8021      IFEND;
       A2  8022      tmv$ptl_p^ [cst_p^.taskid.index].new_task_status := tmc$ts_timeout_reqexp_shortshrt;
       A2  8023      tmv$ptl_p^ [cst_p^.taskid.index].end_of_wait_time := #FREE_RUNNING_CLOCK (0) + tmv$cycle_delay_time;
       C0  8024      cst_p^.dispatch_control.call_dispatcher := TRUE;
       C0  8025      IF (cst_p^.next_ptl0_to_dispatch = 0) THEN
       D6  8026        cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
       DE  8027      IFEND;
       DE  8028
       DE  8029      ?IF tmc$debug_cycle_requests THEN
           8030        IF (osv$debug > 0) THEN
           8031          tmv$cycle_trace [ti].code := tmc$cyc_cause_task_switch;
           8032          tmv$cycle_trace [ti].time := #FREE_RUNNING_CLOCK (0);
           8033          ti := ti + 1;
           8034          IF ti > 10000 THEN
           8035            mtp$error_stop ('TM - trace buffer full');
           8036          IFEND;
           8037        IFEND;
       DE  8038      ?IFEND;
       DE  8039      tmp$clear_lock (tmv$ptl_lock);
      110  8040
```



```
      110  8041    PROCEND tmp$cause_task_switch;
```

TMP$QUEUE_TASK

```
        0  8043
        0  8044   PROCEDURE [XDCL] tmp$queue_task
        0  8045     (     taskid: ost$global_task_id;
        0  8046           task_status: tmt$task_status;
        0  8047      VAR queue_link: tmt$task_queue_link);
        0  8048
        0  8049
        0  8050 {
        0  8051 {  The purpose of this request is to add a task to the
        0  8052 {  end of a task queue.
        0  8053 {
        0  8054 {    TMP$QUEUE_TASK (TASKID, QUEUE_LINK)
        0  8055 {
        0  8056 {  TASKID : (INPUT) This parameter specifies the taskid of
        0  8057 {                   the task to be placed in the queue.
        0  8058 {
        0  8059 {  QUEUE_LINK : (INPUT) This parameter specifies the head of
        0  8060 {                        the task queue.
        0  8061 {
        0  8062
        0  8063     VAR
        0  8064       cst_p: ^ost$cpu_state_table,
        0  8065       ptl_p: ^tmt$primary_task_list_entry;
        0  8066
        0  8067
        0  8068     ptl_p := ^tmv$ptl_p^ [taskid.index];
        4  8069
        4  8070     tmp$set_lock (tmv$ptl_lock);
       50  8071
       50  8072     tmp$stop_if_bad_taskid (taskid);
       B2  8073
       B2  8074     IF (ptl_p^.queue_link.head <> 0) OR (ptl_p^.queue_link.tail <> 0) OR
       D2  8075          (queue_link.tail = taskid.index) THEN
       D2  8076       mtp$error_stop ('TM02 - already queued');
       F2  8077     IFEND;
       F2  8078     IF (task_status < tmc$ts_first_external_queue) OR (ptl_p^.new_task_status > tmc$ts_null) THEN
      104  8079       mtp$error_stop ('TM49 - bad call to queue_task');
      124  8080     IFEND;
      124  8081
      124  8082     IF queue_link.tail = 0 THEN
      130  8083       queue_link.head := taskid.index;
      130  8084       queue_link.tail := taskid.index;
      140  8085     ELSE
      140  8086       tmv$ptl_p^ [queue_link.tail].queue_link.head := taskid.index;
      140  8087       ptl_p^.queue_link.tail := queue_link.tail;
      140  8088       queue_link.tail := taskid.index;
      15C  8089     IFEND;
      15C  8090
      15C  8091     mtp$cst_p (cst_p);
      16C  8092     cst_p^.dispatch_control.call_dispatcher := TRUE;
      16C  8093     IF (cst_p^.next_ptl0_to_dispatch = 0) THEN
      18A  8094       cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
      192  8095     IFEND;
      192  8096     IF taskid <> cst_p^.taskid THEN
      1A6  8097       mtp$error_stop ('TM - attempt to queue task not xtask');
      1C6  8098     IFEND;
```


TMP$QUEUE_TASK

```
      1C6  8099     IF (task_status = tmc$ts_page_wait) OR (task_status = tmc$ts_io_wait_queued) THEN
      1D8  8100       tmv$io_wait_task_count := tmv$io_wait_task_count + 1;
      1E2  8101     IFEND;
      1E2  8102     ptl_p^.new_task_status := task_status;
      1E2  8103
      1E2  8104     tmp$clear_lock (tmv$ptl_lock);
      21E  8105
      21E  8106   PROCEND tmp$queue_task;
        0  8107
```

TMP$DEQUEUE_TASK

```
  0  8109
  0  8110   PROCEDURE [XDCL] tmp$dequeue_task
  0  8111     (VAR queue_link: tmt$task_queue_link;
  0  8112      VAR taskid: ost$global_task_id);
  0  8113
  0  8114
  0  8115 {  The purpose of this procedure is to delink the task at the
  0  8116 {  head of the task queue.
  0  8117 {
  0  8118 {    TMP$DEQUEUE_TASK (QUEUE_LINK, TASKID)
  0  8119 {
  0  8120 {  QUEUE_LINK : (INPUT,OUTPUT) This parameter specifies the head
  0  8121 {                               of the task queue.
  0  8122 {
  0  8123 {  TASKID : (OUTPUT) This parameter specifies the taskid of the
  0  8124 {                    delinked task.
  0  8125 {
  0  8126
  0  8127     VAR
  0  8128       attempt_preselection: boolean,
  0  8129       ijle_p: ^jmt$initiated_job_list_entry,
  0  8130       xcb_p: ^ost$execution_control_block,
  0  8131       ptl_p: ^tmt$primary_task_list_entry;
  0  8132
  0  8133
  0  8134     tmp$set_lock (tmv$ptl_lock);
 42  8135     taskid.index := queue_link.head;
 42  8136     ptl_p := ^tmv$ptl_p^ [taskid.index];
 42  8137     taskid.seqno := ptl_p^.sequence_number;
 42  8138     tmp$stop_if_bad_taskid (taskid);
 C8  8139
 C8  8140     jmp$get_ijle_p (ptl_p^.ijl_ordinal, ijle_p);
 C8  8141
 C8  8142     IF queue_link.head = queue_link.tail THEN
 F6  8143       queue_link.head := 0;
 F6  8144       queue_link.tail := 0;
104  8145     ELSE
104  8146       queue_link.head := ptl_p^.queue_link.head;
104  8147       tmv$ptl_p^ [queue_link.head].queue_link.tail := 0;
104  8148       ptl_p^.queue_link.head := 0;
120  8149     IFEND;
120  8150
120  8151     IF (ptl_p^.status = tmc$ts_page_wait) OR (ptl_p^.status = tmc$ts_io_wait_queued) OR
13E  8152         (ptl_p^.new_task_status = tmc$ts_page_wait) OR (ptl_p^.new_task_status = tmc$ts_io_wait_queued) THEN
13E  8153       tmv$io_wait_task_count := tmv$io_wait_task_count - 1;
148  8154     IFEND;
148  8155
148  8156     IF ptl_p^.status < tmc$ts_first_external_queue THEN
152  8157       IF (ptl_p^.status <> tmc$ts_executing) OR (ptl_p^.new_task_status < tmc$ts_first_external_queue) THEN
160  8158         mtp$error_stop ('TM92 - task not queued');
180  8159       IFEND;
180  8160       ptl_p^.new_task_status := tmc$ts_null;
188  8161     ELSE
188  8162       IF ptl_p^.idle_status = tmc$is_idled THEN
194  8163         ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
194  8164         ptl_p^.status := tmc$ts_ready_but_swapped;
```


TMP$DEQUEUE_TASK

```
194  8165         ptl_p^.idle_status := tmc$is_idled_sched_notified;
194  8166         jmp$ready_task_in_swapped_job (ptl_p^.ijl_ordinal, ijle_p);
1CA  8167       ELSEIF ptl_p^.idle_status < tmc$is_idled THEN
1CE  8168         ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
1CE  8169         ptl_p^.status := tmc$ts_ready;
1CE  8170         xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, ptl_p^.xcb_offset);
1CE  8171         attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
20C  8172         tmp$dct_ready_task (xcb_p, ijle_p, taskid.index, attempt_preselection);
AB8  8173       IFEND;
AB8  8174     IFEND;
AB8  8175
AB8  8176     tmp$clear_lock (tmv$ptl_lock);
AEA  8177
AEA  8178   PROCEND tmp$dequeue_task;
  0  8179
```

```
        o  8181
        o  8182   PROCEDURE remove_task_from_seg_lock_q
        o  8183     (     taskid: ost$global_task_id;
        o  8184           ijle_p: ^jmt$initiated_job_list_entry;
        o  8185           ptl_p: ^tmt$primary_task_list_entry);
        o  8186 {
        o  8187 {  The purpose of this procedure is to remove a task from the segment
        o  8188 {  lock queue. This task has been readied by another task. It will NOT
        o  8189 {  have the segment lock.
        o  8190 {
        o  8191 {    REMOVE_TASK_FROM_SEG_LOCK_Q (taskid, ijle_p, ptl_p)
        o  8192 {
        o  8193 {  TASKID: (INPUT) This parameter specifies the taskid of the
        o  8194 {                  task which is being removed from the segment lock queue.
        o  8195 {
        o  8196 {  IJLE_P: (INPUT) This parameter specifies the ijl pointer of the job
        o  8197 {                  owning the task being removed.
        o  8198 {
        o  8199 {  PTL_P: (INPUT)  This parameter specifies the pointer to the PTL entry
        o  8200 {                  of the task being removed.
        o  8201 {
        o  8202 {  NOTE : Anyone calling this procedure must have tmv$ptl_lock set.
        o  8203 {
        o  8204
        o  8205     VAR
        o  8206       asti: mmt$ast_index,
        o  8207       fde_p: gft$locked_file_desc_entry_p,
        o  8208       found: boolean,
        o  8209       queue_link: tmt$task_queue_link,
        o  8210       queue_link_save: ^tmt$task_queue_link,
        o  8211       sdtx_p: ^mmt$segment_descriptor_extended,
        o  8212       segnum: ost$segment,
        o  8213       task_rb: ^mmt$rb_lock_unlock_segment,
        o  8214       xcb_p: ^ost$execution_control_block;
        o  8215
        o  8216     found := FALSE;
        4  8217     IF ptl_p^.idle_status < tmc$is_idled THEN
       12  8218       xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, ptl_p^.xcb_offset);
       12  8219       task_rb := #LOC (xcb_p^.xp.x_registers [0]);
       32  8220       segnum := #SEGMENT (task_rb^.pva);
       32  8221       sdtx_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
       32  8222       gfp$mtr_get_locked_fde_p (sdtx_p^.sfid, ijle_p, fde_p);
       E2  8223       tmp$remove_task_from_q (ptl_p, fde_p^.segment_lock.task_queue);
      13C  8224       ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      13C  8225       ptl_p^.status := tmc$ts_ready;
      13C  8226       jmv$ajl_p^ [ijle_p^.ajl_ordinal].job_is_good_swap_candidate := FALSE;
*WARN*     8227       tmp$dct_ready_task (xcb_p, ijle_p, taskid.index, TRUE {attempt_preselection});
      B3E  8228     ELSE
      B3E  8229       /search_loop/
      B3E  8230       FOR asti := LOWERBOUND (mmv$ast_p^) TO UPPERBOUND (mmv$ast_p^) DO
      B54  8231         IF (mmv$ast_p^ [asti].in_use) THEN
      B66  8232           gfp$mtr_get_locked_fde_p (mmv$ast_p^ [asti].sfid, ijle_p, fde_p);
      BE8  8233           queue_link := fde_p^.segment_lock.task_queue;
      BE8  8234           WHILE NOT found AND (queue_link.head <> 0) DO
      BFC  8235             IF queue_link.head = taskid.index THEN
      C0A  8236               found := TRUE;
```



```
      C0A  8237               EXIT /search_loop/;
      C18  8238             ELSE
      C18  8239               queue_link := tmv$ptl_p^ [queue_link.head].queue_link;
      C32  8240             IFEND;
      C32  8241           WHILEND;
      C5C  8242         IFEND;
      C5C  8243       FOREND /search_loop/;
      C60  8244
      C60  8245       IF NOT found THEN
      C64  8246         mtp$error_stop ('TM112-task not found in seg lock Q');
      C84  8247       IFEND;
      C84  8248
      C84  8249       tmp$remove_task_from_q (ptl_p, fde_p^.segment_lock.task_queue);
      CDE  8250       ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
      CDE  8251       ptl_p^.status := tmc$ts_ready_but_swapped;
      CDE  8252       ptl_p^.idle_status := tmc$is_idled_sched_notified;
      CDE  8253       jmp$ready_task_in_swapped_job (ptl_p^.ijl_ordinal, ijle_p);
      D14  8254     IFEND;
      D14  8255
      D14  8256   PROCEND remove_task_from_seg_lock_q;
```

TMP$FIND_NEXT_QUEUED_TASK

```
        0  8258
        0  8259    PROCEDURE [XDCL] tmp$find_next_queued_task
        0  8260       (VAR taskid: ost$global_task_id);
        0  8261
        0  8262      VAR
        0  8263        ptl_p: ^tmt$primary_task_list_entry;
        0  8264
        0  8265      tmp$set_lock (tmv$ptl_lock);
       42  8266      tmp$stop_if_bad_taskid (taskid);
       AC  8267      ptl_p := ^tmv$ptl_p^ [taskid.index];
       AC  8268      taskid.index := ptl_p^.queue_link.head;
       AC  8269      taskid.seqno := tmv$ptl_p^ [ptl_p^.queue_link.head].sequence_number;
       AC  8270      tmp$clear_lock (tmv$ptl_lock);
      108  8271
      108  8272    PROCEND tmp$find_next_queued_task;
        0  8273
```


[XDCL] tmp$check_timed_wait_not_queued

```
        0  8275
        0  8276    PROCEDURE [XDCL] tmp$check_timed_wait_not_queued
        0  8277      (    time_next_scan_wait_not_queued: integer);
        0  8278
        0  8279      VAR
        0  8280        ptlo: ost$task_index;
        0  8281
        0  8282      tmp$set_lock (tmv$ptl_lock);
       42  8283
       42  8284      FOR ptlo := 1 TO UPPERBOUND (tmv$ptl_p^) DO
       56  8285        IF (tmv$ptl_p^ [ptlo].status = tmc$ts_timed_wait_not_queued) AND (tmv$ptl_p^ [ptlo].
       74  8286              end_of_wait_time <= time_next_scan_wait_not_queued) THEN
       74  8287          tmv$ptl_p^ [ptlo].status := tmc$ts_timeout_reqexp_longvlong;
       74  8288          tmp$insert_timed_wait_queue (ptlo);
      144  8289        IFEND;
      144  8290      FOREND;
      14A  8291
      14A  8292      tmp$clear_lock (tmv$ptl_lock);
      182  8293
      182  8294    PROCEND tmp$check_timed_wait_not_queued;
        0  8295
```

TMP$INSERT_TIMED_WAIT_QUEUE

```
o  8297
o  8298    PROCEDURE [INLINE] tmp$insert_timed_wait_queue
o  8299      (     ptlo: ost$task_index);
o  8300
o  8301  {
o  8302  {  The purpose of this request is to add a task to
o  8303  {  tmv$timed_wait_queue.  Tasks are arranged in the queue
o  8304  {  in ascending order according to their 'end_of_wait_time'.
o  8305  {
o  8306  {    TMP$INSERT_TIMED_WAIT_QUEUE (PTLO)
o  8307  {
o  8308  {  PTLO : (INPUT) This parameter specifies the ptl ordinal of
o  8309  {                the task to be placed in the queue.
o  8310  {
o  8311  {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
o  8312  {
o  8313
o  8314      VAR
o  8315        next_index: ost$task_index,
o  8316        previous_index: ost$task_index,
o  8317        end_of_wait_time: integer;
o  8318
o  8319      next_index := tmv$timed_wait_queue.head;
o  8320      previous_index := 0;
o  8321      end_of_wait_time := tmv$ptl_p^ [ptlo].end_of_wait_time;
o  8322
o  8323      IF tmv$timed_wait_queue.head = 0 THEN
o  8324        tmv$timed_wait_queue.head := ptlo;
o  8325        tmv$timed_wait_queue.tail := ptlo;
o  8326      ELSE
o  8327        WHILE (tmv$ptl_p^ [next_index].end_of_wait_time <= end_of_wait_time) AND
o  8328              (next_index <> 0) AND (next_index <> ptlo) DO
o  8329          previous_index := next_index;
o  8330          next_index := tmv$ptl_p^ [next_index].queue_link.head;
o  8331        WHILEND;
o  8332
o  8333        IF ptlo = next_index THEN   {Dont put this in the while loop - it kills optimization.}
o  8334          mtp$error_stop ('TM02.5 - already queued');
o  8335        IFEND;
o  8336
o  8337        tmv$ptl_p^ [ptlo].queue_link.head := next_index;
o  8338        IF next_index = tmv$timed_wait_queue.head THEN
o  8339          tmv$timed_wait_queue.head := ptlo;
o  8340        ELSE
o  8341          tmv$ptl_p^ [previous_index].queue_link.head := ptlo;
o  8342        IFEND;
o  8343        tmv$ptl_p^ [ptlo].queue_link.tail := previous_index;
o  8344        IF previous_index = tmv$timed_wait_queue.tail THEN
o  8345          tmv$timed_wait_queue.tail := ptlo;
o  8346        ELSE
o  8347          tmv$ptl_p^ [next_index].queue_link.tail := ptlo;
o  8348        IFEND;
o  8349      IFEND;
o  8350
o  8351    PROCEND tmp$insert_timed_wait_queue;
o  8352
```


[INLINE] tmp$remove_task_from_q

```
o  8354 { Purpose:
o  8355 {   This procedure removes a task from either the timed wait queue, or
o  8356 { a segment lock queue.
o  8357
o  8358    PROCEDURE [INLINE] tmp$remove_task_from_q
o  8359      (     ptle_p: ^tmt$primary_task_list_entry;
o  8360       VAR queue: tmt$task_queue_link);
o  8361
o  8362      IF ptle_p^.queue_link.head = 0 THEN
o  8363        queue.tail := ptle_p^.queue_link.tail;
o  8364      ELSE
o  8365        tmv$ptl_p^ [ptle_p^.queue_link.head].queue_link.tail := ptle_p^.queue_link.tail;
o  8366      IFEND;
o  8367
o  8368      IF ptle_p^.queue_link.tail = 0 THEN
o  8369        queue.head := ptle_p^.queue_link.head;
o  8370      ELSE
o  8371        tmv$ptl_p^ [ptle_p^.queue_link.tail].queue_link.head := ptle_p^.queue_link.head;
o  8372      IFEND;
o  8373
o  8374      ptle_p^.queue_link.head := 0;
o  8375      ptle_p^.queue_link.tail := 0;
o  8376
o  8377    PROCEND tmp$remove_task_from_q;
o  8378
```

TMP$CREATE_JOB

```
0  8380
0  8381   PROCEDURE [XDCL] tmp$create_job
0  8382     (VAR rb: tmt$rb_initiate_job;
0  8383          cst_p: ^ost$cpu_state_table);
0  8384
0  8385
0  8386 {
0  8387 {  This procedure is used to schedule a newly created job monitor
0  8388 {  task.  A PTL entry is assigned to the job monitor task and it
0  8389 {  is inserted in the DCT chain.  The segment table entry for the
0  8390 {  new job's JFS is deleted from the address space of the current
0  8391 {  task that initiated this job.  Execution of the new job may
0  8392 {  actually begin before the requesting task continues execution.
0  8393 {
0  8394 {    TMP$CREATE_JOB (RB)
0  8395 {
0  8396 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
0  8397 {
0  8398
0  8399     VAR
0  8400       attempt_preselection: boolean,
0  8401       sdt_p: mmt$max_sdt_p,
0  8402       ktt: packed record
0  8403         case boolean of
0  8404         = TRUE =
0  8405           s: string (5),
0  8406         = FALSE =
0  8407           f1: 0 .. 0fffff(16),
0  8408           f2: 0 .. 0fff(16),
0  8409         casend,
0  8410       recend,
0  8411       job_segnum: ost$segment,
0  8412       segnum: ost$segment,
0  8413       ijl_ordinal: jmt$ijl_ordinal,
0  8414       ijle_p: ^jmt$initiated_job_list_entry,
0  8415       xcb_p: ^ost$execution_control_block,
0  8416       ajl_ordinal: jmt$ajl_ordinal,
0  8417       jm: ost$keypoint_mask,
0  8418       jcb_p: ^jmt$job_control_block;
0  8419
0  8420
0  8421
0  8422
0  8423 { Make an entry in the monitor segment table for the JOB FIXED SEGMENT of the new job.
0  8424
0  8425     sdt_p := #ADDRESS (1, cst_p^.ajlo + mtc$job_fixed_segment, cst_p^.xcb_p^.sdt_offset);
4  8426     job_segnum := #SEGMENT (rb.xcb_p);
4  8427     jmp$assign_ajl_entry (sdt_p^.st [job_segnum].ste.asid, rb.ijlo, jmc$swapping_ajl,
70  8428           FALSE (must assign), ajl_ordinal, rb.status);
70  8429     IF NOT rb.status.normal THEN
78  8430       RETURN;
7A  8431     IFEND;
7A  8432
7A  8433     rb.ajo := ajl_ordinal;
7A  8434
7A  8435 { Make an entry in the PTL for the new task.
```


TMP$CREATE_JOB

```
7A  8436
7A  8437     segnum := rb.ajo + mtc$job_fixed_segment;
7A  8438     ijl_ordinal := jmv$ajl_p^ [rb.ajo].ijl_ordinal;
7A  8439     jmp$get_ijle_p (ijl_ordinal, ijle_p);
7A  8440     IF syv$all_jobs_selected_for_debug THEN
C4  8441       ijle_p^.system_breakpoint_selected := TRUE;
C8  8442     IFEND;
C8  8443     xcb_p := #ADDRESS (1, segnum, #OFFSET (rb.xcb_p));
C8  8444     tmp$set_lock (tmv$ptl_lock);
114  8445     tmp$assign_ptl (xcb_p, ijl_ordinal, rb.jmtr_taskid, rb.status);
1E8  8446     IF NOT rb.status.normal THEN
1F0  8447       mtv$monitor_segment_table.st [segnum].ste.v1 := osc$v1_invalid_entry;
1F0  8448       #PURGE_BUFFER (osc$purge_all_page_seg_map, null_pva);
21C  8449       jmp$free_ajl_with_lock (ijle_p, jmc$swapping_ajl);
234  8450       rb.ajo := jmc$null_ajl_ordinal;
234  8451       tmp$clear_lock (tmv$ptl_lock);
26E  8452       RETURN;
270  8453     IFEND;
270  8454     xcb_p^.global_task_id := rb.jmtr_taskid;
270  8455     tmv$total_task_count := tmv$total_task_count + 1;
270  8456
270  8457 { Initialize the IJL entry and the JCB for the job.
270  8458
270  8459     jcb_p := #ADDRESS (1, rb.ajo + mtc$job_fixed_segment, 0);
270  8460     jcb_p^.jcb_identifier := 0ff00(16);
270  8461     jcb_p^.job_monitor_id := rb.jmtr_taskid;
270  8462     jcb_p^.last_execution_time := #FREE_RUNNING_CLOCK (0);
2B0  8463     jcb_p^.ijle_p := ijle_p;
2B0  8464     jcb_p^.ijl_ordinal := ijl_ordinal;
2B0  8465     ijle_p^.swap_status := jmc$iss_executing;
2B0  8466     ijle_p^.sfd_p := NIL;
2B0  8467     ijle_p^.job_fixed_contiguous_pages := 0;
2B0  8468     ijle_p^.statistics.ready_task_count := 1;
2B0  8469     ijle_p^.statistics.tasks_not_in_long_wait := 1;
2B0  8470     ijle_p^.job_monitor_taskid := rb.jmtr_taskid;
2B0  8471     jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_in_memory_non_swap);
33C  8472     IF syv$perf_keypoints_enabled.swapping_keypoints THEN
348  8473       ktt.s := ijle_p^.system_supplied_name (16, 4);
356  8474       #KEYPOINT (osk$performance, osk$m * ktt.f1, ptk$new_job_name_1);
366  8475       #KEYPOINT (osk$performance, osk$m * ((ktt.f2 * 256) + rb.ajo), ptk$new_job_name_2);
380  8476     IFEND;
380  8477     jm := $ost$keypoint_mask [];
380  8478     IF (osv$keypoint_control.environment = osc$system_keypoints) OR
394  8479           (osv$keypoint_control.environment = osc$system_sample_keypoints) THEN
394  8480       jm := osv$keypoint_control.jm;
394  8481       xcb_p^.keypoint_register_enable := TRUE;
39C  8482     IFEND;
39C  8483     IF jm <> $ost$keypoint_mask [] THEN
3A4  8484       xcb_p^.xp.flags := xcb_p^.xp.flags + $ost$flags [osc$keypoint_enable];
3BA  8485     ELSE
3BA  8486       xcb_p^.xp.flags := xcb_p^.xp.flags - $ost$flags [osc$keypoint_enable];
3CC  8487     IFEND;
3CC  8488     xcb_p^.xp.keypoint_mask := jm;
3CC  8489
3CC  8490 {  Set up the new job's segment table from calling task's segment table.  The xcb segment in calling
3CC  8491 {  task will be put in the proper place in the new job's segment table.
```

TMP$CREATE_JOB

```
3CC  8492
3CC  8493     mmp$create_job (rb.ajo, job_segnum, cst_p^.xcb_p, xcb_p);
3F4  8494
3F4  8495 { Insert the PTL entry for the new job into the Dispatch Tables.
3F4  8496     xcb_p^.timeslice := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
3F4  8497           dispatching_control [ijle_p^.dispatching_control.dispatching_control_index].dispatching_timeslice;
3F4  8498     attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
430  8499     tmp$dct_ready_task (xcb_p, ijle_p, rb.jmtr_taskid.index, attempt_preselection);
CD6  8500
CD6  8501 { Increment the count of initiated jobs that scheduler checks.  This count must be changed in
CD6  8502 { monitor with the ptl lock set.
CD6  8503
CD6  8504     jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].scheduler_initiated_jobs :=
CD6  8505           jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].
CD6  8506           scheduler_initiated_jobs + 1;
CD6  8507
CD6  8508     tmp$clear_lock (tmv$ptl_lock);
D22  8509
D22  8510   PROCEND tmp$create_job;
```


TMP$EXIT_JOB

```
  0  8512
  0  8513   PROCEDURE [XDCL] tmp$exit_job
  0  8514     (VAR rb: tmt$rb_exit_job;
  0  8515           cst_p: ^ost$cpu_state_table);
  0  8516
  0  8517
  0  8518 {
  0  8519 {  The purpose of this procedure is to exit a job monitor task.
  0  8520 {  The PTL entry for the job monitor task is deleted and the job
  0  8521 {  status in the ajl entry is set to terminated.
  0  8522 {
  0  8523 {    TMP$EXIT_JOB (RB)
  0  8524 {
  0  8525 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
  0  8526 {
  0  8527
  0  8528     VAR
  0  8529       xcb_p: ^ost$execution_control_block;
  0  8530
  0  8531
  0  8532     tmp$set_lock (tmv$ptl_lock);
 42  8533     IF tmv$ptl_p^ [cst_p^.taskid.index].idle_status = tmc$is_idled THEN
 60  8534       mtp$error_stop ('TM04 - swapped job exiting');
 80  8535     IFEND;
 80  8536     IF tmv$ptl_p^ [cst_p^.taskid.index].ijl_thread <> 0 THEN
 94  8537       mtp$error_stop ('TM - exit job with active task(s).');
 B4  8538     IFEND;
 B4  8539
 B4  8540 { Free the PTL entry for the task.
 B4  8541
 B4  8542     tmv$total_task_count := tmv$total_task_count - 1;
 B4  8543     cst_p^.ijle_p^.executing_task_count := cst_p^.ijle_p^.executing_task_count - 1;
 B4  8544     free_ptl (cst_p^.taskid.index);
102  8545
102  8546 { Decrement the job count that scheduler uses.  This count must be changed in monitor
102  8547 { with the ptl lock set.
102  8548
102  8549     jmv$job_counts.service_class_counts [cst_p^.ijle_p^.job_scheduler_data.service_class].
102  8550           scheduler_initiated_jobs := jmv$job_counts.service_class_counts [cst_p^.ijle_p^.
102  8551           job_scheduler_data.service_class].scheduler_initiated_jobs - 1;
102  8552
102  8553     tmp$clear_lock (tmv$ptl_lock);
154  8554
154  8555 { Delete the current task from the ready string and cause a task switch.
154  8556
154  8557     xcb_p := cst_p^.xcb_p;
154  8558     cst_p^.xcb_p := NIL;
154  8559     cst_p^.dispatch_control.call_dispatcher := TRUE;
154  8560     IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
16C  8561       cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
174  8562     IFEND;
174  8563
174  8564 {  Free memory resources of any segments still in jobs address space that are unique to the job.
174  8565 {  Deletes the job fixed segment from monitor's address space too, can no longer reference the job's xcb.
174  8566
174  8567     mmp$exit_job (xcb_p);
```

TMP$EXIT_JOB

```
184  8568
184  8569 { Update the service class statistics for the service class this job belongs to.
184  8570
184  8571     jmp$update_service_class_stats (cst_p^.ijle_p);
198  8572
198  8573 { Free the AJL and notify scheduler that the job has terminated.
198  8574
198  8575     jmp$free_ajl_entry (cst_p^.ijle_p, jmc$swapping_ajl);
1B0  8576     jmp$set_job_terminated (cst_p^.ijl_ordinal, cst_p^.ijle_p);
1CC  8577
1CC  8578   PROCEND tmp$exit_job;
```


TMP$CREATE_TASK

```
  0  8580
  0  8581   PROCEDURE [XDCL] tmp$create_task
  0  8582     (VAR rb: tmt$rb_initiate_task;
  0  8583          cst_p: ^ost$cpu_state_table);
  0  8584
  0  8585
  0  8586 {
  0  8587 {  This procedure is used to schedule a newly created task.
  0  8588 {  A PTL entry is assigned to the task and it is inserted
  0  8589 {  in the DCT chain.  The PTL entry is also linked to the
  0  8590 {  AJL thread for this job.
  0  8591 {
  0  8592 {    TMP$CREATE_TASK (RB)
  0  8593 {
  0  8594 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
  0  8595 {
  0  8596
  0  8597     VAR
  0  8598       attempt_preselection: boolean,
  0  8599       jm: ost$keypoint_mask,
  0  8600       wrb: tmt$rb_delay,
  0  8601       offset: integer,
  0  8602       xcb_p: ^ost$execution_control_block;
  0  8603
  0  8604
  0  8605     offset := #OFFSET (rb.xcb_p);
  4  8606     xcb_p := #ADDRESS (1, cst_p^.ajlo + mtc$job_fixed_segment, offset);
  4  8607     tmp$set_lock (tmv$ptl_lock);
 66  8608     tmp$assign_ptl (xcb_p, cst_p^.ijl_ordinal, rb.taskid, rb.status);
148  8609     IF NOT rb.status.normal THEN
150  8610       tmp$clear_lock (tmv$ptl_lock);
182  8611       RETURN;
184  8612     IFEND;
184  8613
184  8614     IF tmv$ptl_p^ [cst_p^.taskid.index].idle_status = tmc$is_idle_initiated THEN
198  8615       tmv$ptl_p^ [rb.taskid.index].idle_status := tmc$is_idled_sched_notified;
198  8616       tmv$ptl_p^ [rb.taskid.index].status := tmc$ts_ready_but_swapped;
198  8617       jmp$ready_task_in_swapped_job (cst_p^.ijl_ordinal, cst_p^.ijle_p);
1CC  8618     ELSE
1CC  8619       IF cst_p^.ijl_ordinal = jmv$system_ijl_ordinal THEN
1D8  8620         cst_p^.ijle_p^.job_scheduler_data.service_class := 1;
1E0  8621       IFEND;
1E0  8622       xcb_p^.subsystem_lock_priority_count := 0;
1E0  8623       xcb_p^.dispatching_priority := cst_p^.xcb_p^.dispatching_priority;
1E0  8624       xcb_p^.timeslice := jmv$service_classes [cst_p^.ijle_p^.job_scheduler_data.service_class]^.attributes.
1E0  8625             dispatching_control [cst_p^.ijle_p^.dispatching_control.dispatching_control_index].
1E0  8626             dispatching_timeslice;
1E0  8627       attempt_preselection := (cst_p^.ijle_p^.multiprocessing_allowed) OR
22E  8628             (cst_p^.ijle_p^.executing_task_count = 0);
22E  8629       IF tmv$tables_initialized THEN
236  8630         tmp$dct_ready_task (xcb_p, cst_p^.ijle_p, rb.taskid.index, attempt_preselection);
AEA  8631       IFEND;
AEA  8632     IFEND;
AEA  8633
AEA  8634     IF tmv$tables_initialized THEN
AF2  8635       insert_ijl (rb.taskid, cst_p);
```

TMP$CREATE_TASK

```
B2A  8636      ELSE
B2A  8637        tmv$tables_initialized := TRUE;
B2E  8638      IFEND;
B2E  8639
B2E  8640      tmv$total_task_count := tmv$total_task_count + 1;
B2E  8641
B2E  8642      cst_p^.ijle_p^.statistics.ready_task_count := cst_p^.ijle_p^.statistics.ready_task_count + 1;
B2E  8643      cst_p^.ijle_p^.statistics.tasks_not_in_long_wait := cst_p^.ijle_p^.statistics.tasks_not_in_long_wait + 1;
B2E  8644      cst_p^.ijle_p^.task_created_after_last_swap := TRUE;
B2E  8645
B2E  8646      xcb_p^.global_task_id := rb.taskid;
B2E  8647      xcb_p^.parent_global_task_id := cst_p^.taskid;
B2E  8648
B2E  8649  { Each task inherits its parents keypoint_enable flag.
B2E  8650      xcb_p^.keypoint_enable := cst_p^.xcb_p^.keypoint_enable;
B2E  8651
B2E  8652      IF (osv$keypoint_control.environment = osc$system_keypoints) OR
B84  8653          (osv$keypoint_control.environment = osc$system_sample_keypoints) THEN
B84  8654        jm := osv$keypoint_control.jm;
B84  8655        xcb_p^.keypoint_register_enable := TRUE;
B90  8656      ELSE
B90  8657  { check if correct job
B90  8658        IF xcb_p^.keypoint_enable = TRUE THEN
B94  8659  { correct - update masks
B94  8660          jm := osv$keypoint_control.jm;
B94  8661          xcb_p^.keypoint_register_enable := TRUE;
BA0  8662        ELSE
BA0  8663  { different - clear masks
BA0  8664          jm := $ost$keypoint_mask [];
BA0  8665          xcb_p^.keypoint_register_enable := FALSE;
BA6  8666        IFEND;
BA6  8667      IFEND;
BA6  8668      IF jm <> $ost$keypoint_mask [] THEN
BAE  8669        xcb_p^.xp.flags := xcb_p^.xp.flags + $ost$flags [osc$keypoint_enable];
BC6  8670      ELSE
BC6  8671        xcb_p^.xp.flags := xcb_p^.xp.flags - $ost$flags [osc$keypoint_enable];
BDA  8672      IFEND;
BDA  8673      xcb_p^.xp.keypoint_mask := jm;
BDA  8674
BDA  8675      mmp$create_task (cst_p^.xcb_p, xcb_p, cst_p^.ijle_p);
BFE  8676
BFE  8677      tmp$clear_lock (tmv$ptl_lock);
C30  8678
C30  8679    PROCEND tmp$create_task;
```


TMP$TASK_EXIT

```
 0  8681
 0  8682    PROCEDURE [XDCL] tmp$task_exit
 0  8683      (VAR rb: tmt$rb_task_exit;
 0  8684           cst_p: ^ost$cpu_state_table);
 0  8685
 0  8686
 0  8687  {
 0  8688  {  The purpose of this procedure is to exit the current task.
 0  8689  {  The PTL entry for the task is deleted and the parent task
 0  8690  {  is notified of callee termination.
 0  8691  {
 0  8692  {    TMP$TASK_EXIT (RB)
 0  8693  {
 0  8694  {  RB: (INPUT,OUTPUT) This parameter specifies the request block.
 0  8695  {
 0  8696
 0  8697      VAR
 0  8698        osv$trap_task_errors: [XDCL, #GATE] boolean := FALSE,
 0  8699        pit_value: integer;
 0  8700
 0  8701
 0  8702      rb.status.normal := TRUE;
 4  8703      IF rb.parent_global_task_id <> cst_p^.xcb_p^.parent_global_task_id THEN
26  8704        mtp$set_status_abnormal ('TM', tme$invalid_global_taskid, rb.status);
26  8705        RETURN;
3E  8706      IFEND;
3E  8707
3E  8708      IF osv$trap_task_errors AND ((cst_p^.xcb_p^.system_table_lock_count DIV 256) <>
5C  8709          0) THEN
5C  8710        mtp$error_stop (' Task exit with system tables locked ');
7C  8711      IFEND;
7C  8712
7C  8713      tmp$update_system_task_list (cst_p^.xcb_p);
90  8714
90  8715  {  Send signal to the parent task.
90  8716
90  8717      tmp$send_signal (rb.parent_global_task_id, rb.signal, rb.status);
BE  8718      IF NOT rb.status.normal THEN
C6  8719        RETURN;
C8  8720      IFEND;
C8  8721
C8  8722
C8  8723      mmp$exit_task (cst_p^.xcb_p);
DC  8724      IF osv$special_aam_trap AND (cst_p^.xcb_p^.special_trap_count <> 0) THEN
F0  8725        mtp$error_stop (' Trap AAM Lock problem ');
110 8726      IFEND;
110 8727      tmp$set_lock (tmv$ptl_lock);
148 8728      cst_p^.ijle_p^.maxws_aio_slowdown_display := 0;
148 8729      cst_p^.ijle_p^.statistics.ready_task_count := cst_p^.ijle_p^.statistics.ready_task_count - 1;
148 8730      cst_p^.ijle_p^.statistics.tasks_not_in_long_wait := cst_p^.ijle_p^.statistics.tasks_not_in_long_wait - 1;
148 8731      cst_p^.ijle_p^.task_created_after_last_swap := FALSE;
148 8732      cst_p^.accumulated_monitor_cptime := 0ffffffff(16) - #READ_REGISTER (osc$pr_process_interval_timer);
170 8733      pit_value := cst_p^.xcb_p^.xp.process_interval_timer_1 * 10000(16) + cst_p^.xcb_p^.xp.
170 8734           process_interval_timer_2;
170 8735      IF pit_value > 7fffffff(16) THEN
194 8736        pit_value := pit_value - 100000000(16);
```

TMP$TASK_EXIT

```
     19A  8737      IFEND;
     19A  8738      cst_p^.accumulated_job_cptime := cst_p^.accumulated_job_cptime - pit_value;
*WARN*    8739      update_cp_statistics (cst_p);
     640  8740      cst_p^.ijle_p^.executing_task_count := cst_p^.ijle_p^.executing_task_count - 1;
     640  8741      free_ptl (cst_p^.taskid.index);
     682  8742      remove_ijl (cst_p^.taskid, cst_p);
     6BC  8743      cst_p^.xcb_p^.task_has_terminated := TRUE;
     6BC  8744      cst_p^.jcb_p^.last_lpid_for_job := cst_p^.cst_index;
     6BC  8745      cst_p^.xcb_p := NIL;
     6BC  8746      cst_p^.dispatch_control.call_dispatcher := TRUE;
     6BC  8747      IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
     6FC  8748        cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
     704  8749      IFEND;
     704  8750
     704  8751      tmv$total_task_count := tmv$total_task_count - 1;
     704  8752
     704  8753      IF tmv$ptl_p^ [cst_p^.taskid.index].idle_status = tmc$is_idle_initiated THEN
     722  8754        initiate_swap_if_possible (cst_p);
     7A6  8755      IFEND;
     7A6  8756      tmp$clear_lock (tmv$ptl_lock);
     7D8  8757
     7D8  8758    PROCEND tmp$task_exit;
```


TMP$MTR_BEGIN_SYSTEM_ACTIVITY

```
       0  8760
       0  8761    PROCEDURE [XDCL] tmp$mtr_begin_lock_activity
       4  8762      (     xcb_p: ^ost$execution_control_block;
       4  8763            activity: 1 .. 256);
       4  8764
       4  8765      xcb_p^.system_table_lock_count := xcb_p^.system_table_lock_count + activity;
       4  8766
       4  8767    PROCEND tmp$mtr_begin_lock_activity;



       0  8769
       0  8770    PROCEDURE [XDCL] tmp$mtr_end_lock_activity
       0  8771      (     cst_p: ^ost$cpu_state_table;
       0  8772            activity: 1 .. 256;
       0  8773       VAR xcb_p: ^ost$execution_control_block);
       0  8774
       0  8775      VAR
       0  8776        ijle_p: ^jmt$initiated_job_list_entry,
       0  8777        new_scheduling_priority: jmt$dispatching_priority,
       0  8778        ptlo: ost$task_index,
       0  8779        state: tmt$find_next_xcb_state,
       0  8780        status: syt$monitor_status,
       0  8781        temp_xcb_p: ^ost$execution_control_block;
       0  8782
       0  8783      tmp$set_lock (tmv$ptl_lock);
      42  8784
      42  8785 { Debug code--verify the lock count.
      42  8786
      42  8787      IF activity = 256 THEN
      4E  8788        IF xcb_p^.system_table_lock_count < 256 THEN
      5E  8789          mtp$error_stop ('TM--system_table_lock_count 1');
      7E  8790        IFEND;
      82  8791      ELSEIF activity = 1 THEN
      86  8792        IF ((xcb_p^.system_table_lock_count MOD 256) = 0) THEN
      AO  8793          mtp$error_stop ('TM--system_table_lock_count 2');
      CO  8794        IFEND;
      C4  8795      ELSE
      C4  8796        mtp$error_stop ('TM--system_table_lock_count 3');
      E4  8797      IFEND;
      E4  8798
      E4  8799 { End debug code.
      E4  8800
      E4  8801
      E4  8802      xcb_p^.system_table_lock_count := xcb_p^.system_table_lock_count - activity;
      E4  8803      ptlo := cst_p^.taskid.index;
      E4  8804
      E4  8805      IF (xcb_p^.system_give_up_cpu) AND
     116  8806           (xcb_p^.system_table_lock_count < 256) AND
     116  8807           (xcb_p^.system_table_lock_count > 0) THEN
     116  8808
     116  8809 {  Reset the task to it's original priority. If the task has subsystem locks set
     116  8810 {  its correct PTL dispatching priority will be determined when the task is readied.
     116  8811 {  The TMP$DCT_READY_TASK procedure will sort out the priorities.
     116  8812
     116  8813        tmv$ptl_p^ [ptlo].dispatching_priority := xcb_p^.dispatching_priority;
```

```
116  8814        xcb_p^.system_give_up_cpu := FALSE;
116  8815        cst_p^.dispatch_control.call_dispatcher := TRUE;
116  8816        IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
13C  8817          cst_p^.dispatching_priority_integer := tmv$dispatch_priority_integer
14C  8818                [xcb_p^.dispatching_priority];
14C  8819        IFEND;
150  8820      ELSEIF (xcb_p^.system_table_lock_count <= 0) AND
16C  8821        (xcb_p^.system_give_up_cpu OR xcb_p^.subsystem_give_up_cpu) THEN
16C  8822
16C  8823        xcb_p^.subsystem_give_up_cpu := FALSE;
16C  8824        xcb_p^.system_give_up_cpu := FALSE;
16C  8825        cst_p^.dispatch_control.call_dispatcher := TRUE;
16C  8826        IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
186  8827          cst_p^.dispatching_priority_integer := tmv$dispatch_priority_integer
19A  8828                [xcb_p^.dispatching_priority];
19A  8829        IFEND;
19A  8830
19A  8831 {   Reset the task to it's original priority. Also, reset the subsystem lock priority
19A  8832 {   which is maintained in the PTL.
19A  8833
19A  8834        tmv$ptl_p^ [ptlo].dispatching_priority := xcb_p^.dispatching_priority;
19A  8835        tmv$ptl_p^ [ptlo].readying_task_priority := 0;
19A  8836
19A  8837        jmp$get_ijle_p (tmv$ptl_p^ [xcb_p^.global_task_id.index].ijl_ordinal, ijle_p);
19A  8838        new_scheduling_priority := ijle_p^.dispatching_control.dispatching_priority;
19A  8839        tmp$find_next_xcb (tmc$fnx_job, ijle_p, tmv$ptl_p^ [xcb_p^.global_task_id.index].ijl_ordinal,
20E  8840              state, temp_xcb_p);
20E  8841        WHILE temp_xcb_p <> NIL DO
21C  8842          IF tmv$ptl_p^ [temp_xcb_p^.global_task_id.index].readying_task_priority >
234  8843                                  new_scheduling_priority THEN
234  8844            new_scheduling_priority := tmv$ptl_p^ [temp_xcb_p^.global_task_id.index].readying_task_priority;
238  8845          IFEND;
238  8846          tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, state, temp_xcb_p);
268  8847        WHILEND;
27E  8848        ijle_p^.scheduling_dispatching_priority := new_scheduling_priority;
27E  8849        xcb_p^.subsystem_lock_priority_count := 0;
27E  8850        tmv$ptl_p^ [ptlo].ptl_flags.subsystem_locks_set := FALSE;
2A4  8851      IFEND;
2A4  8852
2A4  8853      IF xcb_p^.stlc_allocation THEN
2B4  8854        xcb_p^.stlc_allocation := FALSE;
2B4  8855        tmp$set_monitor_flag (xcb_p^.global_task_id, mmc$mf_segment_mgr_flag, status);
2DA  8856      IFEND;
2DA  8857
2DA  8858      ?IF tmc$debug_cycle_requests THEN
     8859        IF (osv$debug > 0) THEN
     8860          tmp$set_lock (tmv$ptl_lock);
     8861          tmv$cycle_trace [ti].code := tmc$cyc_mtr_end_sys_activity;
     8862          tmv$cycle_trace [ti].time := #FREE_RUNNING_CLOCK (0);
     8863          ti := ti + 1;
     8864          IF ti > 10000 THEN
     8865            mtp$error_stop ('TM - trace buffer full');
     8866          IFEND;
     8867          tmp$clear_lock (tmv$ptl_lock);
     8868        IFEND;
2DA  8869      ?IFEND;
```



```
2DA  8870      tmp$clear_lock (tmv$ptl_lock);
314  8871
314  8872    PROCEND tmp$mtr_end_lock_activity;
```

```
      0  8874
      0  8875   PROCEDURE [XDCL] tmp$cycle
      0  8876     (VAR rb: tmt$rb_cycle;
      0  8877          cst_p: ^ost$cpu_state_table);
      0  8878
      0  8879
      0  8880 {
      0  8881 {  The purpose of this procedure is to cause a task switch to
      0  8882 {  occur.  The CYCLE request is issued by the currently executing
      0  8883 {  task.
      0  8884 {
      0  8885 {    TMP$CYCLE
      0  8886 {
      0  8887
      0  8888     VAR
      0  8889       xcb_p: ^ost$execution_control_block,
      0  8890       ijle_p: ^jmt$initiated_job_list_entry,
      0  8891       state: tmt$find_next_xcb_state,
      0  8892       new_scheduling_priority: jmt$dispatching_priority,
      0  8893       lock_ptlo: ost$task_index,
      0  8894       ptlo: ost$task_index;
      0  8895
      0  8896     ptlo := cst_p^.taskid.index;
      4  8897     tmp$set_lock (tmv$ptl_lock);
     4A  8898
     4A  8899     IF NOT tmv$tables_initialized THEN
     5A  8900     ELSEIF (cst_p^.xcb_p^.system_table_lock_count > 0) AND
     76  8901              (cst_p^.xcb_p^.system_table_lock_count < 256) AND
     76  8902              (cst_p^.xcb_p^.system_give_up_cpu) THEN
     76  8903       cst_p^.xcb_p^.system_give_up_cpu := FALSE;
     80  8904     ELSEIF (cst_p^.xcb_p^.system_table_lock_count <= 0) AND
     9C  8905       (cst_p^.xcb_p^.system_give_up_cpu OR cst_p^.xcb_p^.subsystem_give_up_cpu) THEN
     9C  8906
     9C  8907       cst_p^.xcb_p^.subsystem_give_up_cpu := FALSE;
     9C  8908       cst_p^.xcb_p^.system_give_up_cpu := FALSE;
     9C  8909
     9C  8910 {  Reset the task to it's original priority. Also, reset the subsystem lock priority
     9C  8911 {  which is maintained in the PTL.
     9C  8912
     9C  8913       tmv$ptl_p^ [ptlo].dispatching_priority := cst_p^.xcb_p^.dispatching_priority;
     9C  8914       tmv$ptl_p^ [ptlo].readying_task_priority := 0;
     9C  8915
     9C  8916       new_scheduling_priority := cst_p^.ijle_p^.dispatching_control.dispatching_priority;
     9C  8917       tmp$find_next_xcb (tmc$fnx_job, cst_p^.ijle_p, cst_p^.ijl_ordinal, state, xcb_p);
     F2  8918       WHILE xcb_p <> NIL DO
    100  8919         IF tmv$ptl_p^ [xcb_p^.global_task_id.index].readying_task_priority >
    118  8920                                  new_scheduling_priority THEN
    118  8921           new_scheduling_priority := tmv$ptl_p^ [xcb_p^.global_task_id.index].readying_task_priority;
    11C  8922         IFEND;
    11C  8923         tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, state, xcb_p);
    14C  8924       WHILEND;
    162  8925       cst_p^.ijle_p^.scheduling_dispatching_priority := new_scheduling_priority;
    162  8926       cst_p^.xcb_p^.subsystem_lock_priority_count := 0;
    162  8927       tmv$ptl_p^ [ptlo].ptl_flags.subsystem_locks_set := FALSE;
    18A  8928     ELSEIF tmv$cycle_delay_time > 0 THEN
    192  8929       tmv$ptl_p^ [ptlo].new_task_status := tmc$ts_timeout_reqexp_shortshrt;
```



```
    192  8930       tmv$ptl_p^ [ptlo].end_of_wait_time := #FREE_RUNNING_CLOCK (0) + tmv$cycle_delay_time;
    1AE  8931     IFEND;
    1AE  8932
    1AE  8933     cst_p^.dispatch_control.call_dispatcher := TRUE;
    1AE  8934     IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
    1BA  8935       cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
    1C2  8936     IFEND;
    1C2  8937
    1C2  8938     ?IF tmc$debug_cycle_requests THEN
         8939       IF (osv$debug > 0) THEN
         8940         tmv$cycle_trace [ti].code := rb.code;
         8941         tmv$cycle_trace [ti].p1 := rb.p1;
         8942         tmv$cycle_trace [ti].p2 := rb.p2;
         8943         tmv$cycle_trace [ti].time := #FREE_RUNNING_CLOCK (0);
         8944         tmv$cycle_trace [ti].xtask := ptlo;
         8945         lock_ptlo := rb.lock_value DIV 256;
         8946         tmv$cycle_trace [ti].gtid := lock_ptlo;
         8947         tmv$cycle_trace [ti].status := tmv$ptl_p^ [lock_ptlo].status;
         8948         IF tmv$cycle_trace [ti].status = tmc$ts_page_wait THEN
         8949           jmp$get_ijle_p (tmv$ptl_p^ [lock_ptlo].ijl_ordinal, ijle_p);
         8950           xcb_p := #ADDRESS (1, ijle_p^.ajl_ordinal + mtc$job_fixed_segment, tmv$ptl_p^ [lock_ptlo].
         8951                xcb_offset);
         8952           tmv$cycle_trace [ti].utp := xcb_p^.page_wait_info.pva;
         8953           tmv$cycle_trace [ti].p := xcb_p^.xp.p_register.pva;
         8954         IFEND;
         8955         ti := ti + 1;
         8956         IF ti > 10000 THEN
         8957           mtp$error_stop ('TM - trace buffer full');
         8958         IFEND;
         8959       IFEND;
    1C2  8960     ?IFEND;
    1C2  8961     tmp$clear_lock (tmv$ptl_lock);
    1FA  8962
    1FA  8963   PROCEND tmp$cycle;
```

TMP$DELAY

```
        0  8965
        0  8966   PROCEDURE [XDCL] tmp$delay
        0  8967     (VAR rb: tmt$rb_delay;
        0  8968          cst_p: ^ost$cpu_state_table);
        0  8969
        0  8970
        0  8971 {
        0  8972 {  The purpose of this procedure is to suspend execution of the
        0  8973 {  requesting task until the time specified by the task.  The
        0  8974 {  DELAY request is issued by the currently executing task.
        0  8975 {
        0  8976 {    TMP$DELAY (RB)
        0  8977 {
        0  8978 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
        0  8979 {
        0  8980 {  TYPE
        0  8981 {    TMT$RB_DELAY = RECORD
        0  8982 {      REQCODE,
        0  8983 {      STATUS,
        0  8984 {      REQUESTED_WAIT_TIME,
        0  8985 {      EXPECTED_WAIT_TIME,
        0  8986 {    RECEND;
        0  8987 {
        0  8988 {  REQCODE: (input) The value of this parameter is SYC$RC_DELAY.
        0  8989 {  STATUS: (output) This parameter specifies the system status.
        0  8990 {  REQUESTED_WAIT_TIME: (input) This parameter specifies the requested
        0  8991 {           wakeup time for this task.
        0  8992 {  EXPECTED_WAIT_TIME: (input) This parameter specifies the expected
        0  8993 {           wakeup time for this task.
        0  8994 {
        0  8995
        0  8996     VAR
        0  8997       ptle_p: ^tmt$primary_task_list_entry;
        0  8998
        0  8999     IF NOT tmv$tables_initialized THEN
       10  9000       cst_p^.max_cptime := 20000;
       10  9001       RETURN;
       1E  9002     IFEND;
       1E  9003
       1E  9004     tmp$set_lock (tmv$ptl_lock);
       58  9005     IF mlv$c170_rqst_blk.req <> NIL THEN
       6A  9006       IF (cst_p^.xcb_p^.global_task_id = mlv$c170_rqst_blk.req^.task_id) AND
       8A  9007            (cst_p^.xcb_p^.xp.p_register.pva.ring > 1) THEN
       8A  9008         IF mtv$mli_status.wait_inhibit THEN
       96  9009           cst_p^.xcb_p^.wait_inhibited := TRUE;
       96  9010           tmp$clear_lock (tmv$ptl_lock);
       CE  9011           RETURN;
       D4  9012         ELSE
       D4  9013           mtv$mli_status.ready := FALSE;
       DA  9014         IFEND;
       DA  9015       IFEND;
       DA  9016     IFEND;
       DA  9017
       DA  9018     ptle_p := ^tmv$ptl_p^ [cst_p^.taskid.index];
       DA  9019
       DA  9020     IF ptle_p^.monitor_flags <> $syt$monitor_flags [] THEN
```


TMP$DELAY

```
       F4  9021       cst_p^.xcb_p^.xp.user_condition_register := cst_p^.xcb_p^.xp.
       F4  9022            user_condition_register + $ost$user_conditions [osc$free_flag];
       F4  9023       cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags + ptle_p^.monitor_flags;
       F4  9024       cst_p^.xcb_p^.system_flags := cst_p^.xcb_p^.system_flags + ptle_p^.system_flags;
       F4  9025
       F4  9026       ptle_p^.monitor_flags := $syt$monitor_flags [];
       F4  9027       ptle_p^.system_flags := $tmt$system_flags [];
       F4  9028       tmp$clear_lock (tmv$ptl_lock);
      15A  9029       RETURN;
      15C  9030     IFEND;
      15C  9031
      15C  9032     IF rb.requested_wait_time < UPPERVALUE (ost$free_running_clock) THEN
      16C  9033       IF (rb.expected_wait_time < tmv$long_wait_swap_time) OR (cst_p^.xcb_p^.system_table_lock_count > 0) THEN
      184  9034         ptle_p^.new_task_status := tmc$ts_timeout_reqexp_longlong;
      18E  9035       ELSEIF (rb.expected_wait_time < tmv$timed_wait_not_queued) THEN
      196  9036         ptle_p^.new_task_status := tmc$ts_timeout_reqexp_longvlong;
      1A0  9037       ELSE
      1A0  9038         ptle_p^.new_task_status := tmc$ts_timed_wait_not_queued;
      1A6  9039       IFEND;
      1AA  9040     ELSE
      1AA  9041       IF (rb.expected_wait_time < tmv$long_wait_swap_time) OR (cst_p^.xcb_p^.system_table_lock_count > 0) THEN
      1C2  9042         ptle_p^.new_task_status := tmc$ts_timeout_reqexp_inflong;
      1CC  9043       ELSE
      1CC  9044         ptle_p^.new_task_status := tmc$ts_timeout_reqexp_infvlong;
      1D2  9045       IFEND;
      1D2  9046     IFEND;
      1D2  9047     ptle_p^.end_of_wait_time := rb.requested_wait_time;
      1D2  9048     tmp$clear_lock (tmv$ptl_lock);
      210  9049     cst_p^.dispatch_control.call_dispatcher := TRUE;
      210  9050     IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
      21C  9051       cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
      224  9052     IFEND;
      224  9053
      224  9054
      224  9055   PROCEND tmp$delay;
```

```
      0  9057    PROCEDURE [XDCL] tmp$mtr_wait
      0  9058      (VAR rb (input, output) : tmt$rb_wait_signal;
      0  9059           cst_p: ^ost$cpu_state_table);
      0  9060
      0  9061
      0  9062  {
      0  9063  {  The purpose of this procedure is to process the job mode
      0  9064  {  request to suspend execution of the current task until the
      0  9065  {  specified time has expired or an event has occured.  However
      0  9066  {  execution of the task is not suspended if the WAIT INHIBITED
      0  9067  {  flag is set in the XCB.
      0  9068  {
      0  9069  {    TMP$MTR_WAIT (RB)
      0  9070  {
      0  9071  {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
      0  9072  {
      0  9073  {  TYPE
      0  9074  {    TMT$RB_WAIT_SIGNAL = RECORD
      0  9075  {      REQCODE,
      0  9076  {      STATUS,
      0  9077  {      REQUESTED_WAIT_TIME,
      0  9078  {      EXPECTED_WAIT_TIME,
      0  9079  {    RECEND;
      0  9080  {
      0  9081  {  REQCODE: (input) The value of this parameter is SYC$RC_WAIT.
      0  9082  {  STATUS: (output) This parameter specifies the standard monitor status.
      0  9083  {  REQUESTED_WAIT_TIME: (input) This parameter specifies the maximum amount
      0  9084  {       of time to wait before resuming execution.
      0  9085  {  EXPECTED_WAIT_TIME: (input) This parameter specifies the expected amount
      0  9086  {       of time to wait before resuming execution.
      0  9087  {
      0  9088
      0  9089      VAR
      0  9090        new_task_status: tmt$task_status,
      0  9091        ptle_p: ^tmt$primary_task_list_entry,
      0  9092        readied_ijle_p: ^jmt$initiated_job_list_entry,
      0  9093        readying_task_priority: jmt$dispatching_priority,
      0  9094        service_class_p: ^jmt$service_class_attributes;
      0  9095
      0  9096
      0  9097  { If the request was PMP$READY_TASK_AND_WAIT, the rb.global_taskid contains the taskid of the task
      0  9098  { to be readied.  That taskid must be a valid taskid.
      0  9099
      0  9100      tmp$set_lock (tmv$ptl_lock);
     42  9101      IF rb.global_taskid <> tmv$null_global_task_id THEN
     5A  9102        tmp$check_taskid_with_lock_set (rb.global_taskid, tmc$opt_return, rb.status);
     BC  9103        IF NOT rb.status.normal THEN
     C4  9104          tmp$clear_lock (tmv$ptl_lock);
     F6  9105          RETURN;
     F8  9106        IFEND;
     F8  9107      IFEND;
     F8  9108
     F8  9109
     F8  9110  { Determine the new task status based on the length of time the task wants to wait.
     F8  9111
     F8  9112      IF rb.requested_wait_time < UPPERVALUE (ost$free_running_clock) THEN
```



```
    104  9113        IF (rb.expected_wait_time < tmv$long_wait_swap_time) OR (cst_p^.xcb_p^.system_table_lock_count > 0) THEN
    120  9114          new_task_status := tmc$ts_timeout_reqexp_longlong;
    126  9115        ELSEIF (rb.expected_wait_time < tmv$timed_wait_not_queued) THEN
    12E  9116          new_task_status := tmc$ts_timeout_reqexp_longvlong;
    136  9117        ELSE
    136  9118          new_task_status := tmc$ts_timed_wait_not_queued;
    13A  9119        IFEND;
    13E  9120      ELSE
    13E  9121        IF (rb.expected_wait_time < tmv$long_wait_swap_time) OR (cst_p^.xcb_p^.system_table_lock_count > 0) THEN
    15A  9122          new_task_status := tmc$ts_timeout_reqexp_inflong;
    160  9123        ELSE
    160  9124          new_task_status := tmc$ts_timeout_reqexp_infvlong;
    162  9125        IFEND;
    164  9126      IFEND;
    164  9127
    164  9128
    164  9129  { If this WAIT request will cause the job to go into long wait then do the
    164  9130  { cyclic aging that is done as part of swapout. Note: long wait aging may cause tasks of the job to go ready
    164  9131  { to assign backing files to transient segments. In this case the NEXT long wait should NOT
    164  9132  { do LONG WAIT aging.
    164  9133
    164  9134      ptle_p := ^tmv$ptl_p^ [cst_p^.taskid.index];
    164  9135      ptle_p^.new_task_status := new_task_status;
    164  9136      IF ((new_task_status = tmc$ts_timeout_reqexp_infvlong) OR
    1A8  9137          (new_task_status = tmc$ts_timeout_reqexp_longvlong) OR
    1A8  9138          (new_task_status = tmc$ts_timed_wait_not_queued)) AND
    1A8  9139          (cst_p^.ijle_p^.statistics.tasks_not_in_long_wait = 1) AND NOT
    1A8  9140           cst_p^.ijle_p^.long_wait_aging_complete AND (cst_p^.ajlo > 0) THEN
    1A8  9141        IF (ptle_p^.monitor_flags = $syt$monitor_flags []) AND NOT
    1C8  9142            (cst_p^.xcb_p^.wait_inhibited OR
    1C8  9143            (ptle_p^.ptl_flags.wait_inhibited = tmc$wi_wait_inhibited)) THEN
    1C8  9144          jsp$long_wait_aging (cst_p^.ijle_p);
    1D8  9145          cst_p^.ijle_p^.long_wait_aging_complete := TRUE;
    1E0  9146        IFEND;
    1E0  9147      IFEND;
    1E0  9148
    1E0  9149
    1E0  9150  { If the task has pending monitor flags or has wait inhibited, let it keep running.
    1E0  9151  { If long wait aging has readied this task then PTL.NEW_TASK_STAUS will be null.
    1E0  9152
    1E0  9153      new_task_status := ptle_p^.new_task_status;
    1E0  9154      IF (ptle_p^.monitor_flags <> $syt$monitor_flags []) OR (new_task_status = tmc$ts_null) THEN
    1F0  9155        cst_p^.xcb_p^.xp.user_condition_register := cst_p^.xcb_p^.xp.
    1F0  9156            user_condition_register + $ost$user_conditions [osc$free_flag];
    1F0  9157        cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags + ptle_p^.monitor_flags;
    1F0  9158        cst_p^.xcb_p^.system_flags := cst_p^.xcb_p^.system_flags + ptle_p^.system_flags;
    1F0  9159        ptle_p^.monitor_flags := $syt$monitor_flags [];
    1F0  9160        ptle_p^.system_flags := $tmt$system_flags [];
    1F0  9161        new_task_status := tmc$ts_null;
    22E  9162      ELSEIF cst_p^.xcb_p^.wait_inhibited OR
    246  9163          (ptle_p^.ptl_flags.wait_inhibited =  tmc$wi_wait_inhibited) THEN
    246  9164        new_task_status := tmc$ts_null;
    24E  9165      IFEND;
    24E  9166
    24E  9167      IF new_task_status <> tmc$ts_null THEN
    252  9168        IF cst_p^.xcb_p^.xp.p_register.pva.ring <= osc$tsrv_ring THEN
```

```
264  9169          ptle_p^.ptl_flags.wait_inhibited := tmc$wi_wait_selected_r3;
274  9170        ELSE
274  9171          ptle_p^.ptl_flags.wait_inhibited := tmc$wi_wait_selected;
282  9172        IFEND;
282  9173        ptle_p^.end_of_wait_time := rb.requested_wait_time;
282  9174        cst_p^.dispatch_control.call_dispatcher := TRUE;
282  9175        IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
296  9176          cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
29E  9177        IFEND;
2A2  9178      ELSE
2A2  9179        cst_p^.xcb_p^.wait_inhibited := FALSE;
2A2  9180        ptle_p^.ptl_flags.wait_inhibited := tmc$wi_null;
2A2  9181        ptle_p^.new_task_status := tmc$ts_null;
2BA  9182      IFEND;
2BA  9183
2BA  9184 { If user request was PMP$READY_TASK_AND_WAIT, the rb.global_taskid is the taskid of the task to
2BA  9185 { be readied.
2BA  9186
2BA  9187      IF rb.global_taskid <> tmv$null_global_task_id THEN
2CA  9188        IF cst_p^.xcb_p^.dispatching_priority >=
2E6  9189             tmv$ptl_p^ [cst_p^.xcb_p^.global_task_id.index].readying_task_priority THEN
2E6  9190          readying_task_priority := cst_p^.xcb_p^.dispatching_priority;
2EA  9191        ELSE
2EA  9192          readying_task_priority := tmv$ptl_p^ [cst_p^.xcb_p^.global_task_id.index].readying_task_priority;
2EC  9193        IFEND;
2EC  9194        tmp$set_task_ready (rb.global_taskid, readying_task_priority,
308  9195           tmc$rc_ready_conditional_wi);
308  9196        jmp$get_ijle_p (tmv$ptl_p^ [rb.global_taskid.index].ijl_ordinal, readied_ijle_p);
308  9197        IF readied_ijle_p^.entry_status = jmc$ies_job_swapped THEN
340  9198          service_class_p := ^jmv$service_classes [cst_p^.ijle_p^.job_scheduler_data.service_class]^.attributes;
340  9199          cst_p^.ijle_p^.job_scheduler_data.service_accumulator_since_swap :=
340  9200                service_class_p^.guaranteed_service_quantum;
340  9201          IF readied_ijle_p^.job_scheduler_data.priority > service_class_p^.scheduling_priority.minimum THEN
36C  9202            IF cst_p^.ijle_p^.job_scheduler_data.priority >= readied_ijle_p^.job_scheduler_data.priority THEN
374  9203              cst_p^.ijle_p^.job_scheduler_data.priority := readied_ijle_p^.job_scheduler_data.priority - 1;
37C  9204            IFEND;
380  9205          ELSEIF readied_ijle_p^.job_scheduler_data.priority = service_class_p^.scheduling_priority.minimum THEN
384  9206            cst_p^.ijle_p^.job_scheduler_data.priority := readied_ijle_p^.job_scheduler_data.priority;
388  9207          IFEND;
388  9208        IFEND;
388  9209      IFEND;
388  9210
388  9211      tmp$clear_lock (tmv$ptl_lock);
3BA  9212
3BA  9213
3BA  9214    PROCEND tmp$mtr_wait;
```



```
  0  9216
  0  9217    PROCEDURE [XDCL] tmp$check_for_swapout_candidate
  0  9218      (     ajl_ordinal: jmt$ajl_ordinal);
  0  9219
  0  9220      VAR
  0  9221        ijle_p: ^jmt$initiated_job_list_entry,
  0  9222        next_ready_time: integer,
  0  9223        ptlo: ost$task_index,
  0  9224        xcb_p: ^ost$execution_control_block;
  0  9225
  0  9226
  0  9227      IF (ajl_ordinal = jmv$system_ajl_ordinal) OR NOT jmv$swap_jobs_in_long_wait THEN
1C  9228        RETURN;
1E  9229      IFEND;
1E  9230
1E  9231      ijle_p := jmv$ajl_p^ [ajl_ordinal].ijle_p;
1E  9232      ptlo := ijle_p^.job_monitor_taskid.index;
1E  9233      next_ready_time := 0ffffffffffff(16);
1E  9234
1E  9235 { Do long wait aging if not already done.
1E  9236
1E  9237      IF NOT ijle_p^.long_wait_aging_complete THEN
44  9238        jsp$long_wait_aging (ijle_p);
54  9239        ijle_p^.long_wait_aging_complete := TRUE;
5A  9240      IFEND;
5A  9241
5A  9242 { If the job now has a ready task (that condition must be checked with the PTL lock set) or if a
5A  9243 { task of the job has a system lock set, do NOT swap out the job.
5A  9244 { Scan the PTL thread for the job and search for next ready time.  If the next ready time of all
5A  9245 { tasks is greater than the force-long-wait-swap-time, then swap the job out.
5A  9246
5A  9247      tmp$set_lock (tmv$ptl_lock);
94  9248
94  9249      IF ijle_p^.statistics.ready_task_count > 0 THEN
9C  9250        tmp$clear_lock (tmv$ptl_lock);
D0  9251        RETURN;
D2  9252      IFEND;
D2  9253
D2  9254      WHILE ptlo <> 0 DO
D6  9255        tmp$get_xcb_p_from_ptlo (ptlo, ajl_ordinal, xcb_p);
D6  9256        IF xcb_p^.system_table_lock_count <> 0 THEN
106  9257          tmp$clear_lock (tmv$ptl_lock);
13A  9258          RETURN;
140  9259        ELSEIF (tmv$ptl_p^ [ptlo].status >= tmc$ts_first_status_in_wait_q) AND
15C  9260               (tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_wait_q) AND
15C  9261               (tmv$ptl_p^ [ptlo].end_of_wait_time < next_ready_time) THEN
15C  9262          next_ready_time := tmv$ptl_p^ [ptlo].end_of_wait_time;
160  9263        IFEND;
160  9264        ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
160  9265      WHILEND;
170  9266
170  9267      IF (next_ready_time - #FREE_RUNNING_CLOCK (0)) > tmv$long_wait_force_swap_time THEN
182  9268        tmp$set_swapout_candidate (ajl_ordinal);
192  9269      IFEND;
192  9270
192  9271      tmp$clear_lock (tmv$ptl_lock);
```

TMP$CHECK_FOR_SWAPOUT_CANDIDATE

```
1C4  9272
1C4  9273    PROCEND tmp$check_for_swapout_candidate;
```


TMP$SET_SWAPOUT_CANDIDATE

```
 0  9275
 0  9276 {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
 0  9277
 0  9278    PROCEDURE tmp$set_swapout_candidate
 0  9279      (     ajl_ordinal: jmt$ajl_ordinal);
 0  9280
 0  9281      VAR
 0  9282        temp_next_cyclic_aging: integer,
 0  9283        ptlo: ost$task_index,
 0  9284        jcb_p: ^jmt$job_control_block,
 0  9285        ijle_p: ^jmt$initiated_job_list_entry;
 0  9286
 0  9287      IF (ajl_ordinal = jmv$system_ajl_ordinal) OR NOT jmv$swap_jobs_in_long_wait THEN
18  9288        RETURN;
1A  9289      IFEND;
1A  9290
1A  9291      ijle_p := jmv$ajl_p^ [ajl_ordinal].ijle_p;
1A  9292
1A  9293 { Scan the IJL thread and mark the entries swapped.
1A  9294
1A  9295      ptlo := ijle_p^.job_monitor_taskid.index;
1A  9296
1A  9297      WHILE ptlo <> 0 DO
36  9298        tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled;
36  9299        ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
36  9300      WHILEND;
4A  9301
4A  9302      jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajl_ordinal, 0);
4A  9303      temp_next_cyclic_aging := jcb_p^.next_cyclic_aging_time - #FREE_RUNNING_CLOCK (0);
64  9304      IF temp_next_cyclic_aging < 0 THEN
70  9305        jcb_p^.next_cyclic_aging_time := 0;
78  9306      ELSE
78  9307        jcb_p^.next_cyclic_aging_time := temp_next_cyclic_aging;
7C  9308      IFEND;
7C  9309
7C  9310      jmp$set_swapout_candidate (ajl_ordinal, jmc$sr_long_wait);
92  9311
92  9312    PROCEND tmp$set_swapout_candidate;
 0  9313
```

[XDCL] tmp$idle_non_dispatchable_job

```
        o  9315
        o  9316 { PURPOSE:
        o  9317 {   This procedure will idle a job whose dispatching priority is currently too low to be dispatched.
        o  9318 { DESIGN:
        o  9319 {   This procedure is called from mmp$periodic when a job is discovered to be non-dispatchable.  The job is
        o  9320 {   swapped out only if all tasks can be idled.
        o  9321
        o  9322   PROCEDURE [XDCL] tmp$idle_non_dispatchable_job
        o  9323     (     ajl_ordinal: jmt$ajl_ordinal);
        o  9324
        o  9325     VAR
        o  9326       ijle_p: ^jmt$initiated_job_list_entry,
        o  9327       status: syt$monitor_status;
        o  9328
        o  9329     tmp$set_lock (tmv$ptl_lock);
       42  9330
       42  9331     ijle_p := jmv$ajl_p^ [ajl_ordinal].ijle_p;
       42  9332     IF ijle_p^.entry_status = jmc$ies_job_in_memory THEN
       66  9333       tmp$idle_tasks_in_job (ajl_ordinal, jmc$sr_idle_dispatching, status);
       84  9334       IF status.normal THEN
       8C  9335         jmp$swap_non_dispatchable_job (ajl_ordinal);
       9C  9336       IFEND;
       9C  9337     IFEND;
       9C  9338
       9C  9339     tmp$clear_lock (tmv$ptl_lock);
       DO  9340
       DO  9341   PROCEND tmp$idle_non_dispatchable_job;
        o  9342
```


TMP$IDLE_TASKS_IN_JOB

```
        o  9344
        o  9345 { PURPOSE:
        o  9346 {   The purpose of this procedure is to idle all tasks in a job.
        o  9347 {   NOTE!!! The caller of this procedure MUST set the PTL lock.
        o  9348
        o  9349   PROCEDURE [XDCL] tmp$idle_tasks_in_job
        o  9350     (     ajl_ordinal: jmt$ajl_ordinal;
        o  9351           swapout_reason: jmt$swapout_reasons;
        o  9352      VAR status: syt$monitor_status);
        o  9353
        o  9354     VAR
        o  9355       attempt_preselection: boolean,
        o  9356       end_ptlo: ost$task_index,
        o  9357       ijle_p: ^jmt$initiated_job_list_entry,
        o  9358       jcb_p: ^jmt$job_control_block,
        o  9359       ptlo: ost$task_index,
        o  9360       ready_task_count: integer,
        o  9361       tasks_not_swappable_count: 0 .. osc$max_tasks,
        o  9362       temp_next_cyclic_aging: integer,
        o  9363       xcb_p: ^ost$execution_control_block;
        o  9364
        o  9365     status.normal := TRUE;
        4  9366     ijle_p := jmv$ajl_p^ [ajl_ordinal].ijle_p;
        4  9367
        4  9368     jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajl_ordinal, 0);
        4  9369
        4  9370 { Reject the request if the job is non-swappable.
        4  9371
        4  9372     IF jmv$ajl_p^ [ajl_ordinal].ijle_p^.entry_status = jmc$ies_job_in_memory_non_swap THEN
       44  9373       mtp$set_status_abnormal ('JS', jse$job_executing_non_swappable, status);
       44  9374       RETURN;
       56  9375     IFEND;
       56  9376
       56  9377 { Scan the IJL thread and mark the entries swapped.  If the swapout reason is idle dispatching and ALL
       56  9378 { tasks cannot be idled, then do not idle any tasks.
       56  9379
       56  9380     ptlo := ijle_p^.job_monitor_taskid.index;
       56  9381     ready_task_count := 0;
       56  9382     tasks_not_swappable_count := 0;
       56  9383
       56  9384     WHILE ptlo <> 0 DO
       66  9385       xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
       66  9386       IF (xcb_p^.system_table_lock_count >= osc$system_table_lock_set) OR
       B2  9387           (tmv$ptl_p^ [ptlo].status = tmc$ts_executing) OR
       B2  9388           (tmv$ptl_p^ [ptlo].status = tmc$ts_ready_and_selected) OR
       B2  9389           (xcb_p^.system_table_lock_count > 0) AND ((tmv$ptl_p^ [ptlo].status <
       B2  9390           tmc$ts_timeout_reqexp_longlong) OR (tmv$ptl_p^ [ptlo].status >
       B2  9391           tmc$ts_timeout_reqexp_infvlong)) THEN
       B2  9392         IF swapout_reason <> jmc$sr_idle_dispatching THEN
       BC  9393           tasks_not_swappable_count := tasks_not_swappable_count + 1;
       BC  9394           tmv$ptl_p^ [ptlo].idle_status := tmc$is_idle_initiated;
       CE  9395         ELSE
       CE  9396           end_ptlo := ptlo;
       CE  9397           ptlo := ijle_p^.job_monitor_taskid.index;
       CE  9398           WHILE (ptlo <> end_ptlo) DO
       D8  9399             IF (tmv$ptl_p^ [ptlo].idle_status >= tmc$is_idled) THEN
```

TMP$IDLE_TASKS_IN_JOB

```
      E8  9400                IF tmv$ptl_p^ [ptlo].status = tmc$ts_ready_but_swapped THEN
      F2  9401                  tmv$ptl_p^ [ptlo].status := tmc$ts_ready;
      F6  9402                IFEND;
      F6  9403                IF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
     106  9404                  attempt_preselection := (ijle_p^.multiprocessing_allowed) OR
     11A  9405                        (ijle_p^.executing_task_count = 0);
*WARN*    9406                  tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
     A48  9407                IFEND;
     A48  9408              IFEND;
     A48  9409              tmv$ptl_p^ [ptlo].idle_status := tmc$is_not_idled;
     A48  9410              ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
     A48  9411            WHILEND;
     A5C  9412            mtp$set_status_abnormal ('JS', jse$unable_to_idle_all_tasks, status);
     A5C  9413            RETURN;
     A6E  9414          IFEND;
     A72  9415
     A72  9416        ELSEIF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
     A82  9417          tmp$remove_task_from_dct (ptlo);
     BF0  9418          tmv$ptl_p^ [ptlo].status := tmc$ts_ready_but_swapped;
     BF0  9419          ready_task_count := ready_task_count + 1;
     BF0  9420          tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled_sched_notified;
     C1A  9421        ELSE
     C1A  9422          tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled;
     C1E  9423        IFEND;
     C1E  9424
     C1E  9425        IF xcb_p^.system_table_lock_count > 0 THEN
     C26  9426          tmv$ptl_p^ [ptlo].ptl_flags.subsystem_locks_set := TRUE;
     C36  9427        IFEND;
     C36  9428
     C36  9429        ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
     C36  9430      WHILEND;
     C4E  9431
     C4E  9432      IF tasks_not_swappable_count <> 0 THEN
     C52  9433        mtp$set_status_abnormal ('JS', jse$unable_to_idle_all_tasks, status);
     C64  9434      ELSE
     C64  9435        temp_next_cyclic_aging := jcb_p^.next_cyclic_aging_time - #FREE_RUNNING_CLOCK (0);
     C6A  9436        IF temp_next_cyclic_aging < 0 THEN
     C76  9437          jcb_p^.next_cyclic_aging_time := 0;
     C7C  9438        ELSE
     C7C  9439          jcb_p^.next_cyclic_aging_time := temp_next_cyclic_aging;
     C80  9440        IFEND;
     C80  9441      IFEND;
     C80  9442
     C80  9443    PROCEND tmp$idle_tasks_in_job;
```


TMP$RESTART_IDLED_TASKS

```
       0  9445
       0  9446    PROCEDURE [XDCL] tmp$restart_idled_tasks
       0  9447      (     ajl_ordinal: jmt$ajl_ordinal);
       0  9448
       0  9449      VAR
       0  9450        attempt_preselection: boolean,
       0  9451        cst_p: ^ost$cpu_state_table,
       0  9452        ijle_p: ^jmt$initiated_job_list_entry,
       0  9453        ready_task_count: integer,
       0  9454        xcb_p: ^ost$execution_control_block,
       0  9455        ptlo: ost$task_index;
       0  9456
       0  9457
       0  9458      jmp$get_ijle_p (jmv$ajl_p^ [ajl_ordinal].ijl_ordinal, ijle_p);
       4  9459
       4  9460      IF jmc$dsw_update_keypoint_masks IN ijle_p^.delayed_swapin_work THEN
      4A  9461        osp$update_job_keypoint_mask (ijle_p, jmv$ajl_p^ [ajl_ordinal].ijl_ordinal);
      5E  9462      IFEND;
      5E  9463
      5E  9464      ptlo := ijle_p^.job_monitor_taskid.index;
      5E  9465
      5E  9466  { Scan the IJL thread and mark the entries swapped in.
      5E  9467
      5E  9468      ready_task_count := 0;
      5E  9469      tmp$set_lock (tmv$ptl_lock);
      A4  9470      WHILE ptlo <> 0 DO
      AA  9471        xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
      AA  9472        IF (tmv$ptl_p^ [ptlo].idle_status >= tmc$is_idled) THEN
      D2  9473          IF tmv$ptl_p^ [ptlo].status = tmc$ts_ready_but_swapped THEN
      DC  9474            tmv$ptl_p^ [ptlo].status := tmc$ts_ready;
      DC  9475            ready_task_count := ready_task_count + 1;
      E2  9476          IFEND;
      E2  9477          IF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
      F2  9478            attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*    9479            tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
     A64  9480          IFEND;
     A64  9481        IFEND;
     A64  9482        tmv$ptl_p^ [ptlo].idle_status := tmc$is_not_idled;
     A64  9483        ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
     A64  9484      WHILEND;
     A78  9485
     A78  9486      tmp$clear_lock (tmv$ptl_lock);
     AAE  9487
     AAE  9488    PROCEND tmp$restart_idled_tasks;
```

TMP$IDLE_TASKS_IN_SYSTEM_JOB

```
      0  9490
      0  9491   PROCEDURE tmp$idle_tasks_in_system_job
      0  9492     (    idle_resume_sys_task_kind: tmt$idle_resume_sys_task_kind;
      0  9493      VAR status: syt$monitor_status);
      0  9494
      0  9495     VAR
      0  9496       ptlo: ost$task_index,
      0  9497       xcb_p: ^ost$execution_control_block,
      0  9498       ready_task_count: integer,
      0  9499       ijle_p: ^jmt$initiated_job_list_entry,
      0  9500       tasks_not_swappable_count: 0 .. osc$max_tasks;
      0  9501
      0  9502     status.normal := TRUE;
      4  9503     ijle_p := jmv$ajl_p^ [jmv$system_ajl_ordinal].ijle_p;
      4  9504     tmp$set_lock (tmv$ptl_lock);
     5C  9505
     5C  9506 { Scan the IJL thread and mark the entries swapped.
     5C  9507
     5C  9508     ptlo := ijle_p^.job_monitor_taskid.index;
     5C  9509     ready_task_count := 0;
     5C  9510     tasks_not_swappable_count := 0;
     5C  9511
     5C  9512     IF idle_resume_sys_task_kind = tmc$ir_dm_system_tasks THEN { Idle ONLY the Device_Management tasks. }
     6A  9513
     6A  9514       WHILE ptlo <> 0 DO
     6E  9515         xcb_p := #ADDRESS (1, mtc$job_fixed_segment + jmv$system_ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
     6E  9516         IF (xcb_p^.system_task_id = tmc$stid_administer_log) OR
     A6  9517               (xcb_p^.system_task_id = tmc$stid_dm_split_al) OR
     A6  9518               (xcb_p^.system_task_id = tmc$stid_volume_space_managemnt) THEN
     A6  9519            IF (xcb_p^.system_table_lock_count >= osc$system_table_lock_set) OR
     DA  9520               (tmv$ptl_p^ [ptlo].status = tmc$ts_executing) OR
     DA  9521               (tmv$ptl_p^ [ptlo].status = tmc$ts_ready_and_selected) OR
     DA  9522              (xcb_p^.system_table_lock_count > 0) AND ((tmv$ptl_p^ [ptlo].status <
     DA  9523              tmc$ts_timeout_reqexp_shortshrt)  OR (tmv$ptl_p^ [ptlo].status > tmc$ts_last_status_in_wait_q))
     DA  9524                    THEN
     DA  9525              tasks_not_swappable_count := tasks_not_swappable_count + 1;
     DA  9526              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idle_initiated;
     EC  9527            ELSEIF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
     FC  9528              tmp$remove_task_from_dct (ptlo);
    26A  9529              tmv$ptl_p^ [ptlo].status := tmc$ts_ready_but_swapped;
    26A  9530              ready_task_count := ready_task_count + 1;
    26A  9531              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled_sched_notified;
    294  9532            ELSE
    294  9533              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled;
    298  9534            IFEND;
    298  9535          IFEND;
    298  9536          ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
    298  9537       WHILEND;
    2A8  9538
    2A8  9539       tmp$clear_lock (tmv$ptl_lock);
    2E0  9540
    2E0  9541       IF tasks_not_swappable_count <> 0 THEN
    2E4  9542         { Some Device_Management task is still executing. }
    2E4  9543         mtp$set_status_abnormal ('JS', jse$unable_to_idle_all_tasks, status);
    2F4  9544       IFEND;
    2F6  9545
```


TMP$IDLE_TASKS_IN_SYSTEM_JOB

```
    2F6  9546     ELSE { Idle everything EXCEPT the Device_Management tasks. }
    2F6  9547
    2F6  9548       WHILE ptlo <> 0 DO
    2FA  9549         xcb_p := #ADDRESS (1, mtc$job_fixed_segment + jmv$system_ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
    2FA  9550         IF (xcb_p^.system_task_id <> tmc$stid_administer_log) AND
    332  9551               (xcb_p^.system_task_id <> tmc$stid_dm_split_al) AND
    332  9552               (xcb_p^.system_task_id <> tmc$stid_volume_space_managemnt) THEN
    332  9553            IF (xcb_p^.system_table_lock_count >= osc$system_table_lock_set) OR
    35A  9554              (tmv$ptl_p^ [ptlo].status = tmc$ts_executing) OR
    35A  9555              (tmv$ptl_p^ [ptlo].status = tmc$ts_ready_and_selected) OR
    35A  9556               (xcb_p^.system_table_lock_count > 0) AND ((tmv$ptl_p^ [ptlo].status <
    35A  9557               tmc$ts_timeout_reqexp_shortshrt) OR (tmv$ptl_p^ [ptlo].status > tmc$ts_last_status_in_wait_q))
    35A  9558               THEN
    35A  9559              tasks_not_swappable_count := tasks_not_swappable_count + 1;
    35A  9560              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idle_initiated;
    36C  9561            ELSEIF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
    37C  9562              tmp$remove_task_from_dct (ptlo);
    4EA  9563              tmv$ptl_p^ [ptlo].status := tmc$ts_ready_but_swapped;
    4EA  9564              ready_task_count := ready_task_count + 1;
    4EA  9565              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled_sched_notified;
    514  9566            ELSE
    514  9567              tmv$ptl_p^ [ptlo].idle_status := tmc$is_idled;
    518  9568            IFEND;
    518  9569          IFEND;
    518  9570          ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
    518  9571       WHILEND;
    528  9572
    528  9573       tmp$clear_lock (tmv$ptl_lock);
    560  9574
    560  9575       IF tasks_not_swappable_count <> 1 THEN
    564  9576         { A system task other than the system JOB_MONITOR is still executing. }
    564  9577         mtp$set_status_abnormal ('JS', jse$unable_to_idle_all_tasks, status);
    574  9578       IFEND;
    574  9579     IFEND;
    574  9580
    574  9581   PROCEND tmp$idle_tasks_in_system_job;
```

```
       0  9583
       0  9584    PROCEDURE tmp$restart_tasks_in_system_job
       0  9585      (    idle_resume_sys_task_kind: tmt$idle_resume_sys_task_kind;
       0  9586       VAR status: syt$monitor_status);
       0  9587
       0  9588      VAR
       0  9589        attempt_preselection: boolean,
       0  9590        ptlo: ost$task_index,
       0  9591        ready_task_count: integer,
       0  9592        xcb_p: ^ost$execution_control_block,
       0  9593        ijle_p: ^jmt$initiated_job_list_entry;
       0  9594
       0  9595      status.normal := TRUE;
       4  9596      jmp$get_ijle_p (jmv$ajl_p^ [jmv$system_ajl_ordinal].ijl_ordinal, ijle_p);
       4  9597
       4  9598      IF jmc$dsw_update_keypoint_masks IN ijle_p^.delayed_swapin_work THEN
      4E  9599        osp$update_job_keypoint_mask (ijle_p, jmv$system_ijl_ordinal);
      66  9600      IFEND;
      66  9601
      66  9602      ptlo := ijle_p^.job_monitor_taskid.index;
      66  9603
      66  9604  { Scan the IJL thread and mark the entries swapped in.
      66  9605
      66  9606      ready_task_count := 0;
      66  9607      tmp$set_lock (tmv$ptl_lock);
      AA  9608
      AA  9609      IF idle_resume_sys_task_kind = tmc$ir_dm_system_tasks THEN { Restart ONLY the Device_Management tasks. }
      AE  9610
      AE  9611        WHILE ptlo <> 0 DO
      B2  9612          xcb_p := #ADDRESS (1, mtc$job_fixed_segment + jmv$system_ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
      B2  9613          IF (xcb_p^.system_task_id = tmc$stid_administer_log) OR
      EA  9614                (xcb_p^.system_task_id = tmc$stid_dm_split_al) OR
      EA  9615                (xcb_p^.system_task_id = tmc$stid_volume_space_managemnt) THEN
      EA  9616            IF (tmv$ptl_p^ [ptlo].idle_status >= tmc$is_idled) THEN
      FC  9617              IF tmv$ptl_p^ [ptlo].status = tmc$ts_ready_but_swapped THEN
     106  9618                tmv$ptl_p^ [ptlo].status := tmc$ts_ready;
     106  9619                ready_task_count := ready_task_count + 1;
     10C  9620              IFEND;
     10C  9621              IF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
     11C  9622                attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*    9623                tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
      A52  9624              IFEND;
      A52  9625            IFEND;
      A52  9626            tmv$ptl_p^ [ptlo].idle_status := tmc$is_not_idled;
      A5E  9627          IFEND;
      A5E  9628          ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
      A5E  9629        WHILEND;
      A6E  9630
      A6E  9631        tmp$clear_lock (tmv$ptl_lock);
      AA8  9632
      AA8  9633      ELSE { Restart everything EXCEPT the Device_Management tasks. }
      AA8  9634
      AA8  9635        WHILE ptlo <> 0 DO
      AAC  9636          xcb_p := #ADDRESS (1, mtc$job_fixed_segment + jmv$system_ajl_ordinal, tmv$ptl_p^ [ptlo].xcb_offset);
      AAC  9637          IF (xcb_p^.system_task_id <> tmc$stid_administer_log) AND
      AE4  9638                (xcb_p^.system_task_id <> tmc$stid_dm_split_al) AND
```



```
      AE4  9639                (xcb_p^.system_task_id <> tmc$stid_volume_space_managemnt) THEN
      AE4  9640            IF (tmv$ptl_p^ [ptlo].idle_status >= tmc$is_idled) THEN
      AEE  9641              IF tmv$ptl_p^ [ptlo].status = tmc$ts_ready_but_swapped THEN
      AF8  9642                tmv$ptl_p^ [ptlo].status := tmc$ts_ready;
      AF8  9643                ready_task_count := ready_task_count + 1;
      AFE  9644              IFEND;
      AFE  9645              IF tmv$ptl_p^ [ptlo].status <= tmc$ts_last_status_in_dct THEN
      B0E  9646                attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*    9647                tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
     146A  9648              IFEND;
     146A  9649            IFEND;
     146A  9650            tmv$ptl_p^ [ptlo].idle_status := tmc$is_not_idled;
     1476  9651          IFEND;
     1476  9652          ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
     1476  9653        WHILEND;
     1486  9654
     1486  9655        tmp$clear_lock (tmv$ptl_lock);
     14BC  9656      IFEND;
     14BC  9657
     14BC  9658    PROCEND tmp$restart_tasks_in_system_job;
```

update_cp_statistics

```
o  9660
o  9661   PROCEDURE [INLINE] update_cp_statistics
o  9662     (     cst_p: ^ost$cpu_state_table);
o  9663
o  9664 {
o  9665 {  The purpose of this procedure is to update the cp statistics in the
o  9666 {  IJL, XCB, JCB and the system statistics record.
o  9667 {
o  9668 {    UPDATE_CP_STATISTICS (CST_P)
o  9669 {
o  9670 {  CST_P: (INPUT) This parameter specifies the pointer to the CPU
o  9671 {                 STATE TABLE.
o  9672 {
o  9673
o  9674     VAR
o  9675       excess_cp_time_used: integer,
o  9676       minor_time_slice_remaining: integer,
o  9677       major_time_slice_remaining: integer,
o  9678       status: syt$monitor_status,
o  9679       task_time_slice_used: integer,
o  9680       total_cptime: integer;
o  9681
o  9682
o  9683 { Update cp statistics in IJL.
o  9684
o  9685     cst_p^.ijle_p^.statistics.cp_time.time_spent_in_mtr_mode := cst_p^.ijle_p^.statistics.cp_time.
o  9686           time_spent_in_mtr_mode + cst_p^.accumulated_monitor_cptime;
o  9687     cst_p^.ijle_p^.statistics.cp_time.time_spent_in_job_mode := cst_p^.ijle_p^.statistics.cp_time.
o  9688           time_spent_in_job_mode + cst_p^.accumulated_job_cptime;
o  9689
o  9690     task_time_slice_used := cst_p^.accumulated_job_cptime + cst_p^.accumulated_monitor_cptime;
o  9691
o  9692 { Update the dispatching priority controls.
o  9693
o  9694     update_dispatching_controls (task_time_slice_used, cst_p^.dispatching_priority);
o  9695
o  9696 { Update system data statistics.
o  9697
o  9698     tmv$cpu_execution_statistics [cst_p^.dispatching_priority].time_spent_in_mtr_mode :=
o  9699           tmv$cpu_execution_statistics [cst_p^.dispatching_priority].time_spent_in_mtr_mode +  cst_p^.
o  9700           accumulated_monitor_cptime;
o  9701     tmv$cpu_execution_statistics [cst_p^.dispatching_priority].time_spent_in_job_mode :=
o  9702           tmv$cpu_execution_statistics [cst_p^.dispatching_priority].time_spent_in_job_mode +  cst_p^.
o  9703           accumulated_job_cptime;
o  9704
o  9705 { Update cp statistics in XCB.
o  9706
o  9707     cst_p^.xcb_p^.cp_time.time_spent_in_mtr_mode := cst_p^.xcb_p^.cp_time.time_spent_in_mtr_mode + cst_p^.
o  9708           accumulated_monitor_cptime;
o  9709     cst_p^.xcb_p^.cp_time.time_spent_in_job_mode := cst_p^.xcb_p^.cp_time.time_spent_in_job_mode + cst_p^.
o  9710           accumulated_job_cptime;
o  9711
o  9712     IF task_time_slice_used >= cst_p^.ijle_p^.dispatching_control.service_remaining THEN
o  9713
o  9714 { Reset the dispatching control for the task, based on the service class dispatching controls.
o  9715
```


update_cp_statistics

```
o  9716       excess_cp_time_used := task_time_slice_used - cst_p^.ijle_p^.dispatching_control.service_remaining;
o  9717       tmp$reset_dispatching_control (cst_p^.ijle_p, cst_p^.ijl_ordinal, excess_cp_time_used, TRUE);
o  9718
o  9719     ELSE
o  9720
o  9721 {Calculate the time remaining on this tasks time slice.
o  9722
o  9723       cst_p^.ijle_p^.dispatching_control.service_remaining := cst_p^.ijle_p^.dispatching_control.
o  9724             service_remaining - task_time_slice_used;
o  9725       minor_time_slice_remaining := cst_p^.xcb_p^.timeslice.minor - task_time_slice_used;
o  9726       major_time_slice_remaining := cst_p^.xcb_p^.timeslice.major - task_time_slice_used;
o  9727       IF minor_time_slice_remaining < 0 THEN
o  9728         cst_p^.xcb_p^.timeslice.minor := 0;
o  9729       ELSE
o  9730         cst_p^.xcb_p^.timeslice.minor := minor_time_slice_remaining;
o  9731       IFEND;
o  9732       IF major_time_slice_remaining < 0 THEN
o  9733         cst_p^.xcb_p^.timeslice.major := 0;
o  9734       ELSE
o  9735         cst_p^.xcb_p^.timeslice.major := major_time_slice_remaining;
o  9736       IFEND;
o  9737     IFEND;
o  9738
o  9739 {  Send a flag to the job monitor of the current job if the flag interval
o  9740 {  has expired.
o  9741
o  9742     total_cptime := cst_p^.ijle_p^.statistics.cp_time.time_spent_in_job_mode + cst_p^.ijle_p^.statistics.
o  9743           cp_time.time_spent_in_mtr_mode;
o  9744     IF total_cptime - cst_p^.jcb_p^.cptime_signal_last_sent >= cst_p^.jcb_p^.signal_interval THEN
o  9745       tmp$set_system_flag (cst_p^.jcb_p^.job_monitor_id, avc$monitor_statistics_flag, status);
o  9746       cst_p^.jcb_p^.cptime_signal_last_sent := total_cptime;
o  9747     IFEND;
o  9748
o  9749   PROCEND update_cp_statistics;
o  9750
```

[INLINE] update_dispatching_controls

```
o  9752
o  9753   PROCEDURE [INLINE] update_dispatching_controls
o  9754     (     time_used: integer;
o  9755           dispatching_priority: jmt$dispatching_priority);
o  9756
o  9757     VAR
o  9758       dp: jmt$dispatching_priority;
o  9759
o  9760     IF (tmv$dispatching_controls.controls_defined) AND (dispatching_priority < jmc$priority_p9) THEN
o  9761       IF time_used >= tmv$dispatching_control_time.time_left_in_interval THEN  {RESET THE TABLE}
o  9762         tmv$dispatching_control_time := tmv$dispatching_controls.controls;
o  9763         tmv$dispatching_control_sets.minimums_to_satisfy := tmv$dispatching_controls.minimums_to_satisfy;
o  9764         tmv$dispatching_control_sets.maximums_exceeded := $jmt$dispatching_priority_set [];
o  9765         tmv$dispatching_control_sets.enforce_maximums := $jmt$dispatching_priority_set [];
o  9766         tmp$calculate_dct_priority_int;
o  9767         FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
o  9768           update_priority_integer (dp);
o  9769         FOREND;
o  9770       ELSE
o  9771         tmv$dispatching_control_time.time_left_in_interval := tmv$dispatching_control_time.
o  9772               time_left_in_interval - time_used;
o  9773         IF dispatching_priority <> jmc$null_dispatching_priority THEN
o  9774           IF (jmc$dp_conversion - dispatching_priority) IN tmv$dispatching_control_sets.
o  9775                 minimums_to_satisfy THEN
o  9776             IF time_used >= tmv$dispatching_control_time.dispatching_priority_time
o  9777                   [dispatching_priority].minimum_time THEN
o  9778               tmv$dispatching_control_sets.minimums_to_satisfy := tmv$dispatching_control_sets.
o  9779                     minimums_to_satisfy - $jmt$dispatching_priority_set
o  9780                     [jmc$dp_conversion - dispatching_priority];
o  9781               tmp$calculate_dct_priority_int;
o  9782               update_priority_integer (dispatching_priority);
o  9783             ELSE
o  9784               tmv$dispatching_control_time.dispatching_priority_time [dispatching_priority].minimum_time
o  9785                     := tmv$dispatching_control_time.dispatching_priority_time [dispatching_priority].
o  9786                     minimum_time - time_used;
o  9787             IFEND;
o  9788           IFEND;
o  9789
o  9790           IF ((jmc$dp_conversion - dispatching_priority) IN tmv$dispatching_controls.maximums_defined)
o  9791                 AND NOT ((jmc$dp_conversion - dispatching_priority) IN tmv$dispatching_control_sets.
o  9792                 maximums_exceeded) THEN
o  9793             IF time_used >= tmv$dispatching_control_time.dispatching_priority_time
o  9794                   [dispatching_priority].maximum_time THEN
o  9795               tmv$dispatching_control_sets.maximums_exceeded := tmv$dispatching_control_sets.
o  9796                     maximums_exceeded + $jmt$dispatching_priority_set
o  9797                     [jmc$dp_conversion - dispatching_priority];
o  9798               jmv$idle_dispatching_controls.maximums_exceeded := jmv$idle_dispatching_controls.
o  9799                     maximums_exceeded + $jmt$dispatching_priority_set [jmc$dp_conversion -
o  9800                     dispatching_priority];
o  9801               IF (jmc$dp_conversion - dispatching_priority) IN tmv$dispatching_controls.enforce_maximums THEN
o  9802                 tmv$dispatching_control_sets.enforce_maximums := tmv$dispatching_control_sets.
o  9803                       enforce_maximums + $jmt$dispatching_priority_set
o  9804                       [jmc$dp_conversion - dispatching_priority];
o  9805               IFEND;
o  9806               tmp$calculate_dct_priority_int;
o  9807               update_priority_integer (dispatching_priority);
```


[INLINE] update_dispatching_controls

```
o  9808             ELSE
o  9809               tmv$dispatching_control_time.dispatching_priority_time [dispatching_priority].maximum_time
o  9810                     := tmv$dispatching_control_time.dispatching_priority_time [dispatching_priority].
o  9811                     maximum_time - time_used;
o  9812             IFEND;
o  9813           IFEND;
o  9814         IFEND;
o  9815       IFEND;
o  9816     IFEND;
o  9817
o  9818   PROCEND update_dispatching_controls;
o  9819
```

[INLINE] update_priority_integer

```
o  9821 { PURPOSE:
o  9822 {   The purpose of this procedure is to calculate and change the integer priority value for a specific
o  9823 {   dispatching priority.  This is done only when the priority crosses a minimum allocated or maximum
o  9824 {   allocated dispatching allocation threshold.
o  9825
o  9826   PROCEDURE [INLINE] update_priority_integer
o  9827     (     dp: jmt$dispatching_priority);
o  9828
o  9829     VAR
o  9830       local_set: tmt$dispatching_control_sets;
o  9831
o  9832     local_set := tmv$dispatching_control_sets;
o  9833
o  9834     local_set.ready_tasks := $jmt$dispatching_priority_set [jmc$dp_conversion - dp] -
o  9835           (local_set.enforce_maximums * local_set.maximums_exceeded);
o  9836     local_set.enforce_maximums := $jmt$dispatching_priority_set [];
o  9837     local_set.minimums_to_satisfy := local_set.minimums_to_satisfy * local_set.ready_tasks;
o  9838     local_set.maximums_exceeded := local_set.maximums_exceeded * local_set.ready_tasks;
o  9839     local_set.ready_tasks := local_set.ready_tasks XOR (local_set.minimums_to_satisfy +
o  9840           local_set.maximums_exceeded);
o  9841     #unchecked_conversion (local_set, tmv$dispatch_priority_integer [dp]);
o  9842
o  9843   PROCEND update_priority_integer;
o  9844
```


[XDCL, INLINE] tmp$calculate_dct_priority_int

```
o  9846 { PURPOSE:
o  9847 {   The purpose of this procedure is to calculate the integer priority value for the DCT queues
o  9848 {   with ready tasks.  If there is nothing queued in the DCT, the integer priority will be 0.
o  9849 {   If there are tasks queued, the integer priority represents the highest allocated dispatching
o  9850 {   priority.
o  9851
o  9852   PROCEDURE [XDCL, INLINE] tmp$calculate_dct_priority_int;
4  9853
4  9854     VAR
4  9855       local_set: tmt$dispatching_control_sets;
4  9856
4  9857     local_set := tmv$dispatching_control_sets;
4  9858     local_set.ready_tasks := local_set.ready_tasks - (local_set.enforce_maximums *
4  9859           local_set.maximums_exceeded);
4  9860     local_set.enforce_maximums := $jmt$dispatching_priority_set [];
4  9861     local_set.minimums_to_satisfy := local_set.minimums_to_satisfy * local_set.ready_tasks;
4  9862     local_set.maximums_exceeded := local_set.maximums_exceeded * local_set.ready_tasks;
4  9863     local_set.ready_tasks := local_set.ready_tasks XOR (local_set.minimums_to_satisfy +
4  9864           local_set.maximums_exceeded);
4  9865
4  9866     #unchecked_conversion (local_set, tmv$dct_priority_integer);
4  9867
4  9868   PROCEND tmp$calculate_dct_priority_int;
o  9869
```

INITIATE_SWAP_IF_POSSIBLE

```
o  9871
o  9872 {  NOTE : Anyone calling this routine must have tmv$ptl_lock set.
o  9873
o  9874   PROCEDURE [INLINE] initiate_swap_if_possible
o  9875     (    cst_p: ^ost$cpu_state_table);
o  9876
o  9877     VAR
o  9878       jcb_p: ^jmt$job_control_block,
o  9879       temp_next_cyclic_aging: integer,
o  9880       ptlo: ost$task_index;
o  9881
o  9882     ptlo := cst_p^.ijle_p^.job_monitor_taskid.index;
o  9883     WHILE (ptlo <> 0) AND (tmv$ptl_p^ [ptlo].idle_status >= tmc$is_idled) DO
o  9884       ptlo := tmv$ptl_p^ [ptlo].ijl_thread;
o  9885     WHILEND;
o  9886     IF ptlo = 0 THEN
o  9887       jcb_p := #ADDRESS (1, mtc$job_fixed_segment + cst_p^.ajlo, 0);
o  9888       temp_next_cyclic_aging := jcb_p^.next_cyclic_aging_time - #FREE_RUNNING_CLOCK (0);
o  9889       IF temp_next_cyclic_aging < 0 THEN
o  9890         jcb_p^.next_cyclic_aging_time := 0;
o  9891       ELSE
o  9892         jcb_p^.next_cyclic_aging_time := temp_next_cyclic_aging;
o  9893       IFEND;
o  9894       jsp$idle_tasks_complete (cst_p^.ijl_ordinal);
o  9895     IFEND;
o  9896
o  9897   PROCEND initiate_swap_if_possible;
```


TMP$FETCH_TASK_STATISTICS

```
o  9899
o  9900   PROCEDURE [XDCL] tmp$fetch_task_statistics
o  9901     (VAR rb: tmt$rb_fetch_task_statistics;
o  9902          cst_p: ^ost$cpu_state_table);
o  9903
o  9904
o  9905 {
o  9906 {  The purpose of this request is to return task statistics.
o  9907 {  Currently this request returns the MONITOR MODE time
o  9908 {  and the JOB MODE time for the task.
o  9909 {
o  9910 {     TMP$FETCH_TASK_STATISTICS (RB, CST_P)
o  9911 {
o  9912 {  RB: (INPUT,OUTPUT) This parameter specifies the request block.
o  9913 {
o  9914 {  CST_P: (INPUT) This parameter specifies the pointer to the CST.
o  9915 {
o  9916
o  9917     VAR
o  9918       current_pit_value: integer;
o  9919
o  9920     rb.status.normal := TRUE;
 4  9921     rb.monitor_cptime := cst_p^.xcb_p^.cp_time.time_spent_in_mtr_mode + 0ffffffff(16) -
20  9922          #READ_REGISTER (osc$pr_process_interval_timer);
20  9923
20  9924     current_pit_value := cst_p^.xcb_p^.xp.process_interval_timer_1 * 10000(16) + cst_p^.xcb_p^.xp.
20  9925          process_interval_timer_2;
20  9926     IF current_pit_value > 7fffffff(16) THEN
42  9927       current_pit_value := current_pit_value - 100000000(16);
48  9928     IFEND;
48  9929     rb.job_cptime := cst_p^.accumulated_job_cptime - current_pit_value +
48  9930          cst_p^.xcb_p^.cp_time.time_spent_in_job_mode;
48  9931
48  9932   PROCEND tmp$fetch_task_statistics;
```

[XDCL] tmp$mtr_update_job_task_enviro

```
        0  9935
        0  9936    PROCEDURE [XDCL] tmp$mtr_update_job_task_enviro
        0  9937      (VAR rb: tmt$rb_update_job_task_enviro;
        0  9938           cst_p: ^ost$cpu_state_table);
        0  9939
        0  9940
        0  9941      VAR
        0  9942        current_pit_value: integer,
        0  9943        dispatching_controls: jmt$dispatching_controls,
        0  9944        i: integer,
        0  9945        ijle_p: ^jmt$initiated_job_list_entry,
        0  9946        new_max_ptlo: ost$task_index,
        0  9947        old_max_ptlo: ost$task_index,
        0  9948        service_used: integer;
        0  9949
        0  9950      rb.status.normal := TRUE;
        4  9951      tmp$set_lock (tmv$ptl_lock);
       4A  9952      CASE rb.subcode OF
       A6  9953      = tmc$ujte_xp_register =
       A6  9954        CASE rb.register_id OF
       B2  9955        = osc$pr_process_interval_timer =
       B2  9956          current_pit_value := cst_p^.xcb_p^.xp.process_interval_timer_1 * 10000(16) + cst_p^.xcb_p^.xp.
       B2  9957                process_interval_timer_2;
       B2  9958          IF current_pit_value > 7fffffff(16) THEN
       D0  9959            current_pit_value := current_pit_value - 100000000(16);
       D6  9960          IFEND;
       D6  9961          cst_p^.accumulated_job_cptime := cst_p^.accumulated_job_cptime + rb.pit_value - current_pit_value;
       D6  9962          cst_p^.xcb_p^.pit_count := cst_p^.xcb_p^.pit_count + rb.pit_value - current_pit_value;
       D6  9963          cst_p^.xcb_p^.xp.process_interval_timer_1 := rb.pit_value DIV 10000(16);
       D6  9964          cst_p^.xcb_p^.xp.process_interval_timer_2 := rb.pit_value MOD 10000(16);
      10A  9965        ELSE
      10A  9966        CASEND;
      10E  9967
      10E  9968      = tmc$ujte_dispatching_priority =
      10E  9969        jmp$get_ijle_p (rb.ijl_ordinal, ijle_p);
      10E  9970        IF ijle_p <> NIL THEN
      13E  9971
      13E  9972 { Verify that the same job is still using this ijl ordinal.
      13E  9973
      13E  9974          IF ijle_p^.system_supplied_name <> rb.system_supplied_name THEN
      14C  9975            tmp$clear_lock (tmv$ptl_lock);
      182  9976            mtp$set_status_abnormal ('JM', jme$non_existent_job, rb.status);
      182  9977            RETURN;
      19A  9978          IFEND;
      19A  9979
      19A  9980          CASE rb.request_origin OF
      1D0  9981          = tmc$cpo_operator, tmc$cpo_recovery =
      1D0  9982
      1D0  9983 { A null dispatching priority indicates the operator specified DEFAULT; the system must determine
      1D0  9984 { the correct dispatching control set based on total service the job has used.
      1D0  9985 { If the operator has specified a dispatching priority, assign the job that priority.
      1D0  9986
      1D0  9987 { If the null priority is coming from job recovery, we have completed job recovery and must now
      1D0  9988 { set the dispatching priority back to its original value.  It had been set up to the system job
      1D0  9989 { dispatching priority to guarantee that it would swap in and recover.
      1D0  9990
```


[XDCL] tmp$mtr_update_job_task_enviro

```
      1D0  9991            IF rb.dispatching_control_info.dispatching_priority = jmc$null_dispatching_priority THEN
      1D8  9992
      1D8  9993              IF rb.request_origin = tmc$cpo_operator THEN
      1E0  9994                ijle_p^.dispatching_control.operator_set_dispatching_prio := jmc$null_dispatching_priority;
      1EA  9995              IFEND;
      1EA  9996              calculate_service_used (ijle_p, service_used);
      1FE  9997              tmp$reset_dispatching_control (ijle_p, rb.ijl_ordinal, service_used, FALSE);
      22C  9998
      22C  9999            ELSE
      22C 10000              ijle_p^.dispatching_control.operator_set_dispatching_prio := rb.dispatching_control_info.
      22C 10001                    dispatching_priority;
      22C 10002              ijle_p^.dispatching_control.dispatching_priority := rb.dispatching_control_info.
      22C 10003                    dispatching_priority;
      22C 10004              ijle_p^.dispatching_control.service_remaining := jmc$dc_maximum_service_limit;
      22C 10005              tmp$update_job_task_environment (ijle_p, rb.ijl_ordinal, tmc$fnx_job);
      262 10006            IFEND;
      262 10007
      262 10008            IF (ijle_p^.dispatching_control.dispatching_priority <>
      272 10009                              ijle_p^.scheduling_dispatching_priority)  THEN
      272 10010              ijle_p^.scheduling_dispatching_priority := ijle_p^.dispatching_control.dispatching_priority;
      276 10011            IFEND;
      27A 10012
      27A 10013          = tmc$cpo_user =
      27A 10014
      27A 10015 { Store the user requested dispatching priority, determine which priority is allowed, and update.
      27A 10016
      27A 10017            ijle_p^.dispatching_control.user_requested_dispatching_prio := rb.dispatching_control_info.
      27A 10018                  dispatching_priority;
      27A 10019            determine_dispatching_priority (ijle_p);
      298 10020            tmp$update_job_task_environment (ijle_p, rb.ijl_ordinal, tmc$fnx_job);
      2BC 10021
      2BC 10022          = tmc$cpo_set_job_unswappable =
      2BC 10023
      2BC 10024 { A job is set unswappable during job termination.  The job is given system dispatching priority
      2BC 10025 { with unlimited service.
      2BC 10026
      2BC 10027            ijle_p^.dispatching_control.dispatching_control_index := jmc$min_dispatching_control;
      2BC 10028            ijle_p^.scheduling_dispatching_priority := jmc$priority_system_job;
      2BC 10029            ijle_p^.dispatching_control.dispatching_priority := jmc$priority_system_job;
      2BC 10030            ijle_p^.dispatching_control.service_remaining := jmc$dc_maximum_service_limit;
      2BC 10031            ijle_p^.dispatching_control.operator_set_dispatching_prio := jmc$null_dispatching_priority;
      2BC 10032            ijle_p^.dispatching_control.user_requested_dispatching_prio := jmc$null_dispatching_priority;
      2BC 10033            tmp$update_job_task_environment (ijle_p, rb.ijl_ordinal, tmc$fnx_job);
      302 10034
      302 10035          = tmc$cpo_interactive_command, tmc$cpo_save_swap_file_sfid  =
      302 10036
      302 10037 { A new command line has been scanned for an interactive job or the job is through critical initiation;
      302 10038 { the first dispatching control set for the class is used.
      302 10039
      302 10040            IF rb.request_origin = tmc$cpo_interactive_command THEN
      30C 10041              ijle_p^.interactive_task_gtid := tmv$null_global_task_id;
      318 10042            IFEND;
      318 10043            ijle_p^.dispatching_control.dispatching_control_index := jmc$min_dispatching_control;
      318 10044            ijle_p^.dispatching_control.service_remaining := jmv$service_classes
      318 10045                  [ijle_p^.job_scheduler_data.service_class]^.attributes.dispatching_control [1].service_limit;
      318 10046            determine_dispatching_priority (ijle_p);
```

[XDCL] tmp$mtr_update_job_task_enviro

```
34C 10047            tmp$update_job_task_environment (ijle_p, rb.ijl_ordinal, tmc$fnx_job);
372 10048
372 10049          = tmc$cpo_interrupt_restore  =
372 10050
372 10051 { Restore the job's dispatching control information to the values saved from before a user interrupt.
372 10052 { If the dispatching_control_index is no longer a valid one for the job's service class (the service class
372 10053 { has been changed since the dispatching control info was saved), calculate the correct dispatching_
372 10054 { control_index based on the CP service the job has used.
372 10055
372 10056            dispatching_controls := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
39E 10057                  dispatching_control [rb.dispatching_control_info.dispatching_control_index];
39E 10058            IF (NOT dispatching_controls.set_defined) OR (dispatching_controls.service_limit <
3BE 10059                  rb.dispatching_control_info.service_remaining) OR
3BE 10060                  (dispatching_controls.dispatching_priority <> rb.dispatching_control_info.
3BE 10061                  dispatching_priority) THEN
3BE 10062
3BE 10063              calculate_service_used (ijle_p, service_used);
3D6 10064              tmp$reset_dispatching_control (ijle_p, rb.ijl_ordinal, service_used, FALSE);
404 10065            ELSE
404 10066              ijle_p^.dispatching_control.dispatching_control_index := rb.dispatching_control_info.
404 10067                    dispatching_control_index;
404 10068              ijle_p^.dispatching_control.service_remaining := rb.dispatching_control_info.
404 10069                    service_remaining;
404 10070              determine_dispatching_priority (ijle_p);
418 10071              tmp$update_job_task_environment (ijle_p, rb.ijl_ordinal, tmc$fnx_job);
43A 10072            IFEND;
440 10073          ELSE
442 10074
442 10075          CASEND;
442 10076
442 10077        IFEND;
446 10078
446 10079      = tmc$ujte_set_non_swappable =
446 10080        IF tmv$ptl_p^ [cst_p^.taskid.index].idle_status = tmc$is_not_idled THEN
462 10081          jmp$change_ijl_entry_status (cst_p^.ijle_p, jmc$ies_job_in_memory_non_swap);
4B8 10082        ELSE
4B8 10083          tmp$reissue_monitor_request;
4C0 10084          cst_p^.dispatch_control.call_dispatcher := TRUE;
4C4 10085        IFEND;
4C8 10086
4C8 10087      = tmc$ujte_idle_other_sys_tasks =
4C8 10088        tmp$idle_tasks_in_system_job (tmc$ir_other_system_tasks, rb.status);
4DE 10089
4DE 10090      = tmc$ujte_restart_other_systasks =
4DE 10091        tmp$restart_tasks_in_system_job (tmc$ir_other_system_tasks, rb.status);
4F4 10092
4F4 10093      = tmc$ujte_idle_dm_sys_tasks =
4F4 10094        tmp$idle_tasks_in_system_job (tmc$ir_dm_system_tasks, rb.status);
50C 10095
50C 10096      = tmc$ujte_restart_dm_systasks =
50C 10097        tmp$restart_tasks_in_system_job (tmc$ir_dm_system_tasks, rb.status);
524 10098
524 10099      = tmc$ujte_expand_ptl =
524 10100
524 10101 {    Copy the entries from the old PTL to the new PTL.  Then link the new PTL entries into the free queue.
524 10102 {    Reset tmv$ptl_p to point to the new expanded ptl.
```


[XDCL] tmp$mtr_update_job_task_enviro

```
524 10103
524 10104        old_max_ptlo := UPPERBOUND (tmv$ptl_p^);
524 10105        new_max_ptlo := UPPERBOUND (rb.ptl_p^);
524 10106        I#MOVE (tmv$ptl_p, rb.ptl_p, #SIZE (tmv$ptl_p^));
570 10107        FOR i := old_max_ptlo + 1 TO new_max_ptlo - 1 DO
57C 10108          rb.ptl_p^[i].ptl_thread := i + 1;
57C 10109        FOREND;
590 10110        rb.ptl_p^[new_max_ptlo].ptl_thread := 0;
590 10111        tmv$ptl_p := rb.ptl_p;
5A8 10112
5A8 10113        IF tmv$dct [jmc$null_dispatching_priority].queue_head = 0 THEN
5B0 10114          tmv$dct [jmc$null_dispatching_priority].queue_head := old_max_ptlo + 1;
5BC 10115        ELSE
5BC 10116          tmv$ptl_p^ [tmv$dct [jmc$null_dispatching_priority].queue_tail].ptl_thread := old_max_ptlo + 1;
5CE 10117        IFEND;
5CE 10118        tmv$dct [jmc$null_dispatching_priority].queue_tail := new_max_ptlo;
5D6 10119
5D6 10120      = tmc$ujte_update_debug_masks =
5D6 10121        tmp$update_debug_registers;
5E2 10122
5E2 10123      = tmc$ujte_set_task_terminating =
5E2 10124
5E2 10125 { The free_flag is not set here because immediately after the return to job mode,
5E2 10126 { all of the outstanding signals and flags are processed.
5E2 10127
5E2 10128        cst_p^.xcb_p^.task_is_terminating := TRUE;
5E2 10129        cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags + tmv$ptl_p^
5E2 10130              [cst_p^.taskid.index].monitor_flags;
5E2 10131        cst_p^.xcb_p^.system_flags := cst_p^.xcb_p^.system_flags + tmv$ptl_p^
5E2 10132              [cst_p^.taskid.index].system_flags;
5E2 10133        tmv$ptl_p^ [cst_p^.taskid.index].system_flags := $tmt$system_flags [ ];
5E2 10134        tmv$ptl_p^ [cst_p^.taskid.index].monitor_flags := $syt$monitor_flags [ ];
628 10135      ELSE
628 10136
628 10137        tmp$clear_lock (tmv$ptl_lock);
662 10138        mtp$error_stop ('TM - unimplemented subcode');
682 10139
682 10140      CASEND;
682 10141      tmp$clear_lock (tmv$ptl_lock);
6BA 10142
6BA 10143    PROCEND tmp$mtr_update_job_task_enviro;
  0 10144
```

TMP$UPDATE_JOB_TASK_ENVIRONMENT

```
       0 10146
       0 10147 { PURPOSE:
       0 10148 {   This procedure changes the dispatching priority and timeslice in the XCB
       0 10149 {   for all tasks of a job.
       0 10150 { DESIGN:
       0 10151 {   The caller has stored the new dispatching priority in the IJL.
       0 10152 {   If the XCB of the job can be referenced (ie, the job is NOT swapped out)
       0 10153 {   changes to the XCB are made.  If the job is swapped out, the delayed
       0 10154 {   swapin work bit is set, which causes this procedure to be called again
       0 10155 {   when the job swaps back in.
       0 10156
       0 10157   PROCEDURE [XDCL] tmp$update_job_task_environment
       0 10158     (     ijle_p: ^jmt$initiated_job_list_entry;
       0 10159           ijl_ordinal: jmt$ijl_ordinal;
       0 10160           xcb_search: tmt$fnx_search_type);
       0 10161
       0 10162     VAR
       0 10163       attempt_preselection: boolean,
       0 10164       ptlo: ost$task_index,
       0 10165       ptle_p: ^tmt$primary_task_list_entry,
       0 10166       xcb_p: ^ost$execution_control_block,
       0 10167       dispatching_priority: integer, {must be integer}
       0 10168       state: tmt$find_next_xcb_state;
       0 10169
       0 10170     tmp$set_lock (tmv$ptl_lock);
      42 10171     tmp$find_next_xcb (xcb_search, ijle_p, ijl_ordinal, state, xcb_p);
      76 10172     IF xcb_p = NIL THEN
      84 10173       ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work +
      9A 10174             $jmt$delayed_swapin_work [jmc$dsw_update_job_task_enviro];
      9A 10175     ELSE
      9A 10176       WHILE xcb_p <> NIL DO
      A0 10177         IF xcb_p^.dispatching_priority_bias_id <> jmc$dpb_absolute THEN
      AC 10178           IF xcb_p^.dispatching_priority_bias_id = jmc$dpb_positive THEN
      B0 10179             dispatching_priority := ijle_p^.dispatching_control.dispatching_priority
      C2 10180                   + xcb_p^.dispatching_priority_bias;
      C2 10181           ELSE
      C2 10182             dispatching_priority := ijle_p^.dispatching_control.dispatching_priority
      D2 10183                   - xcb_p^.dispatching_priority_bias;
      D2 10184           IFEND;
      D2 10185           IF dispatching_priority > jmc$max_dispatching_priority THEN
      D8 10186             dispatching_priority := jmc$max_dispatching_priority;
      DE 10187           ELSEIF dispatching_priority < jmc$min_dispatching_priority THEN
      E2 10188             dispatching_priority := jmc$min_dispatching_priority;
      E8 10189           IFEND;
      E8 10190           xcb_p^.dispatching_priority := dispatching_priority;
      E8 10191           ptlo := xcb_p^.global_task_id.index;
      E8 10192           ptle_p := ^tmv$ptl_p^ [ptlo];
      E8 10193           xcb_p^.timeslice := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
      E8 10194                 dispatching_control [ijle_p^.dispatching_control.dispatching_control_index].
      E8 10195                 dispatching_timeslice;
      E8 10196           IF xcb_p^.system_table_lock_count <= 0 THEN
     130 10197             IF ptle_p^.status <= tmc$ts_last_status_in_dct THEN
     138 10198               tmp$remove_task_from_dct (ptlo);
     2AA 10199               ptle_p^.dispatching_priority := xcb_p^.dispatching_priority;
     2AA 10200               attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
*WARN*   10201               tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
```


TMP$UPDATE_JOB_TASK_ENVIRONMENT

```
     C46 10202             ELSE
     C46 10203               ptle_p^.dispatching_priority := xcb_p^.dispatching_priority;
     C4E 10204             IFEND;
     C52 10205           IFEND;
     C52 10206         IFEND;
     C52 10207         tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, state, xcb_p);
     C84 10208       WHILEND;
     C98 10209     IFEND;
     C98 10210     tmp$clear_lock (tmv$ptl_lock);
     CD4 10211
     CD4 10212   PROCEND tmp$update_job_task_environment;
       0 10213
```

TMP$UPDATE_JOB_TASK_CPU_SELECTS

```
        0 10215
        0 10216 { PURPOSE:
        0 10217 {   This procedure changes the processor selections the XCB for all tasks of a job.
        0 10218 { DESIGN:
        0 10219 {   The caller has stored the value of the processors which are still logically on in the SCB.  If the XCB of
        0 10220 {   the job can be referenced (ie, the job is NOT swapped out) changes to the XCB are made.  If the job is
        0 10221 {   swapped out, the delayed swapin work bit is set, which causes the job mode version of this procedure to be
        0 10222 {   called when the job swaps back in.
        0 10223
        0 10224   PROCEDURE [XDCL] tmp$update_job_task_cpu_selects
        0 10225     (     ijle_p: ^jmt$initiated_job_list_entry;
        0 10226           ijl_ordinal: jmt$ijl_ordinal;
        0 10227           xcb_search: tmt$fnx_search_type);
        0 10228
        0 10229     VAR
        0 10230       xcb_p: ^ost$execution_control_block,
        0 10231       state: tmt$find_next_xcb_state;
        0 10232
        0 10233     tmp$set_lock (tmv$ptl_lock);
       42 10234     tmp$find_next_xcb (xcb_search, ijle_p, ijl_ordinal, state, xcb_p);
       76 10235     IF xcb_p = NIL THEN
       84 10236       ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work +
       9A 10237             $jmt$delayed_swapin_work [jmc$dsw_adjust_cpu_selections];
       9A 10238     ELSE
       9A 10239       WHILE xcb_p <> NIL DO
       A0 10240         IF xcb_p^.processor_selections * mtv$scb.processors_logically_on = $ost$processor_id_set [ ] THEN
       B6 10241           xcb_p^.processor_selections := mtv$scb.processors_logically_on;
       BA 10242         IFEND;
       BA 10243         tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, state, xcb_p);
       EA 10244       WHILEND;
      100 10245     IFEND;
      100 10246     tmp$clear_lock (tmv$ptl_lock);
      13C 10247
      13C 10248   PROCEND tmp$update_job_task_cpu_selects;
        0 10249
```


CALCULATE_SERVICE_USED

```
        0 10251
        0 10252 { PURPOSE:
        0 10253 {   This procedure calculates the amount of CP service a job has used.  The procedure
        0 10254 {   tmp$reset_dispatching_control determines the dispatching control index to use, based
        0 10255 {   on the SERVICE_USED returned.
        0 10256 { DESIGN:
        0 10257 {   For interactive jobs, return 0 for service_used; the dispatching control index should
        0 10258 {   be reset to the first index.  For batch classes, base service_used on total cp time.
        0 10259 {   If the job's service class has circular dynamic dispatching priorities defined,
        0 10260 {   MOD the service_used by the sums of the service limits.
        0 10261
        0 10262   PROCEDURE calculate_service_used
        0 10263     (     ijle_p: ^jmt$initiated_job_list_entry;
        0 10264      VAR service_used: integer);
        0 10265
        0 10266     VAR
        0 10267       circular_service: integer,
        0 10268       dispatching_control_p: ^jmt$dispatching_control,
        0 10269       dispatching_control_index: jmt$dispatching_control_index;
        0 10270
        0 10271     IF ijle_p^.job_mode <> jmc$batch THEN
        C 10272       service_used := 0;
       14 10273     ELSE
       14 10274       service_used := ijle_p^.statistics.cp_time.time_spent_in_job_mode + ijle_p^.statistics.
       14 10275             cp_time.time_spent_in_mtr_mode - ijle_p^.dispatching_control.cp_service_at_class_switch;
       14 10276       circular_service := 0;
       14 10277       dispatching_control_p := ^jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
       14 10278             dispatching_control;
       14 10279     /calculate_circular_service/
       14 10280       FOR dispatching_control_index := jmc$max_dispatching_control DOWNTO
       4C 10281             jmc$min_dispatching_control DO
       4C 10282         IF dispatching_control_p^ [dispatching_control_index].set_defined THEN
       5C 10283           IF dispatching_control_p^ [dispatching_control_index].service_limit <>
       6A 10284                 jmc$dc_maximum_service_limit THEN
       6A 10285             circular_service := circular_service + dispatching_control_p^ [dispatching_control_index].
       70 10286                   service_limit;
       70 10287           ELSE
       70 10288             EXIT /calculate_circular_service/;
       74 10289           IFEND;
       74 10290         IFEND;
       74 10291       FOREND /calculate_circular_service/;
       7A 10292       IF circular_service <> 0 THEN
       7E 10293         service_used := service_used MOD circular_service;
       8E 10294       IFEND;
       8E 10295     IFEND;
       8E 10296
       8E 10297   PROCEND calculate_service_used;
        0 10298
```

DETERMINE_DISPATCHING_PRIORITY

```
        0 10300
        0 10301 { PURPOSE:
        0 10302 {   This procedure determines which dispatching priority a job should have
        0 10303 {   and stores it in the IJL.
        0 10304 { DESIGN:
        0 10305 {   If the operator has changed the dispatching priority for a job, that dispatching
        0 10306 {   priority is used.  Otherwise the job is assigned the lesser of the user requested
        0 10307 {   dispatching priority (if there is one) and the dispatching priority defined in the
        0 10308 {   service class table dispatching control sets.
        0 10309
        0 10310   PROCEDURE determine_dispatching_priority
        0 10311     (   ijle_p: ^jmt$initiated_job_list_entry);
        0 10312
        0 10313     VAR
        0 10314       update_scheduling_priority: boolean;
        0 10315
        0 10316 { The scheduling_dispatching_priority and the dispatching_controls.dispatching_priority will only
        0 10317 { be different in the case of a task of the job having subsystem locks set and being readied by
        0 10318 { a task which has a higher dispatching priority. In that instance, the scheduling_dispatching_priority
        0 10319 { will be updated when the task releases the lock and issues a CYCLE request.
        0 10320
        0 10321     update_scheduling_priority := (ijle_p^.dispatching_control.dispatching_priority = ijle_p^.
        4 10322           scheduling_dispatching_priority);
        4 10323     IF ijle_p^.dispatching_control.operator_set_dispatching_prio <> jmc$null_dispatching_priority THEN
       1C 10324       ijle_p^.dispatching_control.dispatching_priority := ijle_p^.dispatching_control.
       24 10325           operator_set_dispatching_prio;
       24 10326     ELSE
       24 10327       IF (ijle_p^.dispatching_control.user_requested_dispatching_prio <> jmc$null_dispatching_priority) AND
       50 10328           (ijle_p^.dispatching_control.user_requested_dispatching_prio <
       50 10329           jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.
       50 10330           attributes.dispatching_control [ijle_p^.dispatching_control.dispatching_control_index].
       50 10331           dispatching_priority) THEN
       50 10332         ijle_p^.dispatching_control.dispatching_priority := ijle_p^.dispatching_control.
       58 10333             user_requested_dispatching_prio;
       58 10334       ELSE
       58 10335         ijle_p^.dispatching_control.dispatching_priority :=
       7C 10336             jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.
       7C 10337             attributes.dispatching_control [ijle_p^.dispatching_control.dispatching_control_index].
       7C 10338             dispatching_priority;
       7C 10339       IFEND;
       7C 10340     IFEND;
       7C 10341
       7C 10342     IF update_scheduling_priority THEN
       80 10343       ijle_p^.scheduling_dispatching_priority := ijle_p^.dispatching_control.dispatching_priority;
       88 10344     IFEND;
       88 10345
       88 10346   PROCEND determine_dispatching_priority;
        0 10347
```


TMP$RESET_DISPATCHING_CONTROLS

```
        0 10349
        0 10350 { PURPOSE:
        0 10351 {   This procedure is called to reset the dispatching control set for a job.
        0 10352 { DESIGN:
        0 10353 {   If expired_dispatching_control is TRUE, the procedure has been called because a
        0 10354 {   a task switch determined that the job has used all the service allowed for the
        0 10355 {   current dispatching control index.  The next index needs to be assigned.
        0 10356 {   Otherwise the procedure needs to determine the correct index based on
        0 10357 {   the total amount of service the job has used.
        0 10358 {   The procedure finds the correct dispatching control index, determines the
        0 10359 {   dispatching priority, and makes a call to update the dispatching priority
        0 10360 {   in the XCB.
        0 10361
        0 10362   PROCEDURE [XDCL] tmp$reset_dispatching_control
        0 10363     (     ijle_p: ^jmt$initiated_job_list_entry;
        0 10364           ijl_ordinal: jmt$ijl_ordinal;
        0 10365           excess_service_used: integer;
        0 10366           expired_dispatching_control: boolean);
        0 10367
        0 10368     VAR
        0 10369       dispatching_control_p: ^jmt$dispatching_control,
        0 10370       next_index: integer,
        0 10371       service_used: integer;
        0 10372
        0 10373     dispatching_control_p := ^jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
        4 10374           dispatching_control;
        4 10375
        4 10376     service_used := excess_service_used;
        4 10377
        4 10378     IF expired_dispatching_control THEN
       2C 10379       next_index := ijle_p^.dispatching_control.dispatching_control_index + 1;
       36 10380     ELSE
       36 10381       next_index := jmc$min_dispatching_control;
       38 10382     IFEND;
       38 10383
       38 10384   /find_dispatching_control_set/
       38 10385     WHILE service_used >= 0 DO
       3C 10386       IF (next_index <= jmc$max_dispatching_control) AND (dispatching_control_p^ [next_index].set_defined)
       52 10387             THEN
       52 10388         IF dispatching_control_p^ [next_index].service_limit > service_used THEN
       5C 10389           ijle_p^.dispatching_control.service_remaining := dispatching_control_p^ [next_index].service_limit -
       5C 10390                 service_used;
       5C 10391           ijle_p^.dispatching_control.dispatching_control_index := next_index;
       5C 10392           determine_dispatching_priority (ijle_p);
       7A 10393           tmp$update_job_task_environment (ijle_p, ijl_ordinal, tmc$fnx_job);
       9A 10394           EXIT /find_dispatching_control_set/;
       A0 10395         ELSE
       A0 10396           service_used := service_used - dispatching_control_p^ [next_index].service_limit;
       A0 10397           next_index := next_index + 1;
       A4 10398         IFEND;
       A8 10399       ELSE
       A8 10400         next_index := jmc$min_dispatching_control;
       AC 10401       IFEND;
       AC 10402     WHILEND /find_dispatching_control_set/;
       B0 10403
       B0 10404   PROCEND tmp$reset_dispatching_control;
```

TMP$RESET_DISPATCHING_CONTROLS

```
        O 10405
```


TMP$SWITCH_TASK

```
        O 10407
        O 10408    PROCEDURE [XDCL] tmp$switch_task
        O 10409      (     dummy: ^cell;
        O 10410            cst_p: ^ost$cpu_state_table);
        O 10411
        O 10412
        O 10413 {
        O 10414 {  The purpose of this procedure is to change the current task
        O 10415 {  of execution.  The XCB, JCB and AJL accounting fields are
        O 10416 {  updated.
        O 10417 {
        O 10418 {     TMP$SWITCH_TASK
        O 10419 {
        O 10420
        O 10421
        O 10422      VAR
        O 10423        attempt_preselection: boolean,
        O 10424        locked_dp_integer: integer,
        O 10425        next_ptlo_to_dispatch: ost$task_index,
        O 10426        next_task_ijle_p: ^jmt$initiated_job_list_entry,
        O 10427        next_task_xcb_p: ^ost$execution_control_block,
        O 10428        priority: dp_trick_conversion,
        O 10429        local_set: tmt$dispatching_control_sets,
        O 10430        integer_dp_sets: integer,
        O 10431        r: real,
        O 10432        idle_time: integer,
        O 10433        time: integer,
        O 10434        xcb_p: ^ost$execution_control_block,
        O 10435        ptle_p: ^tmt$primary_task_list_entry,
        O 10436        ptlo: ost$task_index;
        O 10437
        O 10438
        O 10439 { Update task accounting info.
        O 10440
        O 10441      #KEYPOINT (osk$mtr, osk$monitor_multiplier * 0, tmk$switch_task);
        E 10442
        E 10443      IF NOT tmv$tables_initialized THEN
       1A 10444        cst_p^.max_cptime := 20000;
       1A 10445        RETURN;
       28 10446      IFEND;
       28 10447
       28 10448 { Wake up memory link helper if 170 has set a flag to activate it.
       28 10449
       28 10450     ?IF NOT DEBUG THEN
       28 10451        IF mtv$mli_status.wait_inhibit AND NOT mtv$mli_status.ready THEN
       3C 10452          tmp$set_task_ready (mlv$c170_rqst_blk.req^.task_id, 0 {readying_task_priority},
       62 10453              tmc$rc_ready_conditional_wi);
       62 10454        IFEND;
       62 10455     ?IFEND;
       62 10456
       62 10457 { If the CPU is not idle (cst_p^.xcb_p is not NIL), update the executing tasks's status.
       62 10458
       62 10459      xcb_p := cst_p^.xcb_p;
       62 10460
       62 10461      tmp$set_lock (tmv$ptl_lock);
       A4 10462
```

TMP$SWITCH_TASK

```
      A4 10463      IF xcb_p <> NIL THEN
      AA 10464
      AA 10465        IF cst_p^.accumulated_job_cptime <> 0 THEN
*WARN*   10466          update_cp_statistics (cst_p);
     56A 10467        IFEND;
     56A 10468
     56A 10469        xcb_p^.last_lpid_for_task := cst_p^.cst_index;
     56A 10470        cst_p^.jcb_p^.last_lpid_for_job := cst_p^.cst_index;
     56A 10471        cst_p^.jcb_p^.last_execution_time := #FREE_RUNNING_CLOCK (0);
     582 10472        ptlo := cst_p^.taskid.index;
     582 10473        ptle_p := ^tmv$ptl_p^ [ptlo];
     582 10474
     582 10475 {  If the task has system tables locked and is still ready, let it keep running.
     582 10476
     582 10477        /system_locks_set/
     582 10478        BEGIN
     582 10479          IF xcb_p^.system_table_lock_count > 0 THEN
     59E 10480            IF (ptle_p^.new_task_status = tmc$ts_null) AND (cst_p^.cpu_state.next_state <> osc$cpu_stepped) AND
     5B6 10481                (cst_p^.next_processor_state = cmc$on) THEN
     5B6 10482              IF (xcb_p^.system_table_lock_count >= osc$system_table_lock_set) THEN
     5BE 10483                cst_p^.dispatching_priority := jmc$prior_system_tbls_locked;
     5BE 10484                locked_dp_integer := tmv$dispatch_priority_integer [jmc$prior_system_tbls_locked];
     5BE 10485                xcb_p^.system_give_up_cpu := TRUE;
     5D0 10486              ELSEIF (xcb_p^.subsystem_lock_priority_count <= tmv$subsystem_prior_threshold) THEN
     5DC 10487                xcb_p^.subsystem_lock_priority_count := xcb_p^.subsystem_lock_priority_count + 1;
     5DC 10488                cst_p^.dispatching_priority := jmc$prior_subsystem_tbls_locked;
     5DC 10489                locked_dp_integer := tmv$dispatch_priority_integer [jmc$prior_subsystem_tbls_locked];
     5DC 10490                xcb_p^.subsystem_give_up_cpu := TRUE;
     5F6 10491              ELSE
     5F6 10492                EXIT /system_locks_set/;
     5FA 10493              IFEND;
     5FA 10494              cst_p^.dual_state_prior_subpriority := tmv$dual_state_dispatch_prior
     5FA 10495                  [cst_p^.dispatching_priority];
     5FA 10496              cst_p^.max_cptime := jmv$service_classes [cst_p^.ijle_p^.job_scheduler_data.service_class]^.
     5FA 10497                    attributes.dispatching_control [cst_p^.ijle_p^.dispatching_control.
     5FA 10498                    dispatching_control_index].dispatching_timeslice.minor;
     5FA 10499
     5FA 10500 { If this processor was being interrupted to switch to a higher priority task, the task must be inserted
     5FA 10501 { in the DCT because this processor cannot be interrupted now.  Do not change the dispatching priority
     5FA 10502 { integer to the locks-set value until after checking/removing a pre-selected task.  When the task was
     5FA 10503 { pre-selcted the dispatching priority integer for the CPU was changed to the pre-selected task's priority.
     5FA 10504 { If it is changed, the task may be pre_selected for this processor again.
     5FA 10505 { Example:  A P6 task is executing; a P10 task has been pre_selected (so dispatching priority integer
     5FA 10506 { is P10).  The P6 task has sub-system locks set, so its priority is raised to P9.  Changing the dispatching
     5FA 10507 { priority integer to P9 before calling dct_ready_task again for the pre-selected P10 task could cause
     5FA 10508 { this CPU to be pre-selected again.
     5FA 10509
     5FA 10510              IF cst_p^.next_ptlo_to_dispatch <> 0 THEN
     63A 10511                next_ptlo_to_dispatch := cst_p^.next_ptlo_to_dispatch;
     63A 10512                jmp$get_ijle_p (tmv$ptl_p^ [next_ptlo_to_dispatch].ijl_ordinal, next_task_ijle_p);
     63A 10513                next_task_ijle_p^.executing_task_count := next_task_ijle_p^.executing_task_count - 1;
     63A 10514                tmv$ptl_p^ [next_ptlo_to_dispatch].status := tmc$ts_ready;
     63A 10515                tmp$get_xcb_p_from_ptlo (next_ptlo_to_dispatch, next_task_ijle_p^.ajl_ordinal,
     63A 10516                      next_task_xcb_p);
     63A 10517                cst_p^.next_ptlo_to_dispatch := 0;
*WARN*   10518                tmp$dct_ready_task (next_task_xcb_p, next_task_ijle_p, next_ptlo_to_dispatch,
```


TMP$SWITCH_TASK

```
    1042 10519                      {attempt_preselection :} TRUE);
    1042 10520              IFEND;
    1042 10521              cst_p^.dispatching_priority_integer := locked_dp_integer;
    1042 10522
    1042 10523              tmp$clear_lock (tmv$ptl_lock);
    107E 10524              #KEYPOINT (osk$mtr, osk$monitor_multiplier * ptlo, tmk$switch_task + osk$m);
    1086 10525              RETURN; {<--- }
    1088 10526            IFEND;
    108C 10527          ELSEIF (ptle_p^.idle_status = tmc$is_idle_initiated) THEN
    1094 10528            ptle_p^.idle_status := tmc$is_idled;
    1094 10529            cst_p^.xcb_p := NIL;
    1094 10530            initiate_swap_if_possible (cst_p);
    1124 10531          IFEND;
    1124 10532        END /system_locks_set/;
    1124 10533
    1124 10534 { The current task will be removed from the CPU. Decrement this task's executing_task_
    1124 10535 { count.  If a task has not been pre_selected for this CPU, set dispatching_priority_integer
    1124 10536 { equal to the highest priority task in the DCT for the pre-select code in
    1124 10537 { dct_ready_task (which will be called if tasks are readied from the timed wait queue).
    1124 10538
    1124 10539        cst_p^.ijle_p^.executing_task_count := cst_p^.ijle_p^.executing_task_count - 1;
    1124 10540        IF (cst_p^.next_ptlo_to_dispatch = 0) THEN
    113A 10541          cst_p^.dispatching_priority_integer := tmv$dct_priority_integer;
    1142 10542        IFEND;
    1148 10543      IFEND; {xcb_p <> NIL}
    1148 10544
    1148 10545 { Update the timed wait queue.  This must be done before considering the status of the task
    1148 10546 { that is currently executing (if there is one); tasks being readied from the timed wait queue
    1148 10547 { should be considered for pre-selection (in tmp$dct_ready_task) before the task that is
    1148 10548 { currently executing.
    1148 10549
*WARN*   10550      update_timed_wait_queue;
    1BFA 10551
    1BFA 10552 { If there is a task executing, check its new status.
    1BFA 10553
    1BFA 10554      IF xcb_p <> NIL THEN
    1BFE 10555
    1BFE 10556        IF ptle_p^.new_task_status > tmc$ts_last_status_in_dct THEN
    1C06 10557          ptle_p^.status := ptle_p^.new_task_status;
    1C06 10558          ptle_p^.new_task_status := tmc$ts_null;
    1C06 10559          cst_p^.ijle_p^.statistics.ready_task_count := cst_p^.ijle_p^.statistics.ready_task_count - 1;
    1C06 10560          IF ptle_p^.status <= tmc$ts_last_status_in_wait_q THEN
    1C26 10561            tmp$insert_timed_wait_queue (ptlo);
    1CF6 10562          IFEND;
    1CF6 10563          IF (ptle_p^.status = tmc$ts_timeout_reqexp_longvlong) OR (ptle_p^.status =
    1D0C 10564                tmc$ts_timed_wait_not_queued) OR (ptle_p^.status = tmc$ts_timeout_reqexp_infvlong) THEN
    1D0C 10565            cst_p^.ijle_p^.statistics.tasks_not_in_long_wait := cst_p^.ijle_p^.statistics.
    1D0C 10566                  tasks_not_in_long_wait - 1;
    1D0C 10567            IF (cst_p^.ijle_p^.statistics.tasks_not_in_long_wait = 0) AND
    1D30 10568                  (cst_p^.ijle_p^.entry_status = jmc$ies_job_in_memory) AND
    1D30 10569                  (ptle_p^.idle_status = tmc$is_not_idled) THEN
    1D30 10570              cst_p^.xcb_p := NIL;
    1D30 10571              tmp$set_swapout_candidate (cst_p^.ajlo);
    1D44 10572            IFEND;
    1D44 10573          IFEND;
    1D48 10574
```

```
1D48 10575        ELSEIF ptle_p^.idle_status <> tmc$is_idled THEN
1D52 10576          ptle_p^.status := tmc$ts_ready;
1D52 10577          attempt_preselection := (cst_p^.next_ptlo_to_dispatch = 0) AND
1D82 10578              ((cst_p^.ijle_p^.multiprocessing_allowed) OR (cst_p^.ijle_p^.statistics.ready_task_count = 1));
1D82 10579          tmp$dct_ready_task (xcb_p, cst_p^.ijle_p, ptlo, attempt_preselection);
264C 10580        ELSE
264C 10581          ptle_p^.status := tmc$ts_ready_but_swapped;
264C 10582          ptle_p^.idle_status := tmc$is_idled_sched_notified;
264C 10583          jmp$ready_task_in_swapped_job (cst_p^.ijl_ordinal, cst_p^.ijle_p);
2678 10584        IFEND;
2678 10585      IFEND;
2678 10586
2678 10587      IF (cst_p^.cpu_state.next_state = osc$cpu_stepped) THEN
2684 10588        ptlo := 0;
2684 10589        tmp$clear_lock (tmv$ptl_lock);
26BE 10590        mtp$error_stop ('terminated through the dispatcher path');
26DE 10591        { Any processor which follows this path is NOT the one driving the system into STEP mode.  THEREFORE,
26DE 10592        { the call above is valid because the driving processor has the correct reason for calling STEP.
26DE 10593        tmp$set_lock (tmv$ptl_lock);
2716 10594      ELSEIF (cst_p^.processor_state <> cst_p^.next_processor_state) OR ((tmv$dedicate_a_cpu_to_nos AND
2742 10595          (cst_p^.dual_state_jps <> 0) AND (cst_p^.next_ptlo_to_dispatch = 0))
2742 10596           AND tmv$multiple_cpus_active) THEN
2742 10597        cst_p^.dual_state_prior_subpriority.subpriority := 0;
2742 10598        ptlo := 0;
274C 10599      ELSE
274C 10600
274C 10601  { Select next task for execution.
274C 10602  { If cst_p^.next_ptlo_to_dispatch is non-zero, that task has already been selected to
274C 10603  { execute on this CPU through the ready task path, so switch to it.
274C 10604  { If the next_ptlo_to_dispatch field is zero, a new task to execute needs to be selected now.
274C 10605  { Tmv$dispatching_control_sets contains the following dispatching priority sets:
274C 10606  { enforce_maximums--priorities which cannot use more than the specified maximum % of the CPU.
274C 10607  { minimums_to_satisfy--priorities which have not received the specified minimum % of the CPU.
274C 10608  { (System priorities can use 100% of the CPU, so they are always in minimums_to_satisfy.)
274C 10609  { ready_tasks--priorities with tasks in the DCT.
274C 10610  { maximums_exceeded--priorities that have exceeded the specified maximum % of the CPU.
274C 10611  { For select_next_task, priorities which have exceeded maximum and for which the maximum is
274C 10612  { enforced are removed from the sets; they cannot execute for the rest of the interval.
274C 10613  { The remaining priorities with ready tasks are manipulated so that each priority is set
274C 10614  { only once in any of the three sets (which are ordered, left to right) : minimums_to_satisfy,
274C 10615  { ready_tasks, maximums_exceeded.  The conversion process pulls off the leftmost bit set in the sets.
274C 10616  { Thus, priorities with a minimum percentage to satisfy are selected first, followed by priorities
274C 10617  { with ready tasks that have not exceeded the maximum percentage.  Priorities which have exceeded
274C 10618  { the maximum but are not prevented from executing by enforce maximum are selected last.
274C 10619  { When the dispatching priority has been selected, tasks from that DCT queue are considered,
274C 10620  { until one is found that can execute.
274C 10621  {
274C 10622  { NOTE:  --PURGE BUFFERS--  If the task or the job the task belongs to did not last execute
274C 10623  { on this CPU, then cache must be purged.  If multiprocessing is allowed, the last processor
274C 10624  { for the task is checked.
274C 10625  { For non-multiprocessing jobs, the last processor for the job is checked.  The task can
274C 10626  { execute only if the job has no other tasks executing.  If a task has been pre_selected to execute,
274C 10627  { the executing_task_count was checked and incremented by the pre-selection code.
274C 10628  { If the task is being selected from the DCT, the task is selected and cache purged if the
274C 10629  { executing_task_count = 0.
274C 10630
```



```
274C 10631        IF cst_p^.next_ptlo_to_dispatch <> 0 THEN
2754 10632          ptlo := cst_p^.next_ptlo_to_dispatch;
2754 10633          ptle_p := ^tmv$ptl_p^ [ptlo];
2754 10634          cst_p^.next_ptlo_to_dispatch := 0;
2754 10635          cst_p^.ijl_ordinal := ptle_p^.ijl_ordinal;
2754 10636          jmp$get_ijle_p (cst_p^.ijl_ordinal, cst_p^.ijle_p);
2754 10637          cst_p^.ajlo := cst_p^.ijle_p^.ajl_ordinal;
2754 10638          cst_p^.jcb_p := #ADDRESS (1, cst_p^.ajlo + mtc$job_fixed_segment, 0);
2754 10639          xcb_p := #ADDRESS (1, mtc$job_fixed_segment + cst_p^.ajlo, ptle_p^.xcb_offset);
2754 10640          IF osv$cpus_logically_on > 1 THEN
27D0 10641            IF cst_p^.ijle_p^.multiprocessing_allowed THEN
27D8 10642              IF xcb_p^.last_lpid_for_task <> cst_p^.cst_index THEN
27E6 10643                { NOTE: The second parameter on the #purge_buffer command is just a dummy pointer.
27E6 10644                #PURGE_BUFFER (osc$purge_all_cache, xcb_p);
27E8 10645                #PURGE_BUFFER (osc$purge_all_page_seg_map, xcb_p);
27EA 10646              IFEND;
27EE 10647            ELSE
27EE 10648              IF cst_p^.jcb_p^.last_lpid_for_job <> cst_p^.cst_index THEN
27FA 10649                #PURGE_BUFFER (osc$purge_all_cache, xcb_p);
27FC 10650                #PURGE_BUFFER (osc$purge_all_page_seg_map, xcb_p);
27FE 10651              IFEND;
2800 10652            IFEND;
2800 10653          IFEND;
2804 10654
2804 10655        ELSE  { Select a new task to execute. }
2804 10656
2804 10657          local_set := tmv$dispatching_control_sets;
2804 10658          local_set.ready_tasks := local_set.ready_tasks - (local_set.enforce_maximums *
2804 10659              local_set.maximums_exceeded);
2804 10660          local_set.enforce_maximums := $jmt$dispatching_priority_set [];
2804 10661          local_set.minimums_to_satisfy := local_set.minimums_to_satisfy * local_set.ready_tasks;
2804 10662          local_set.maximums_exceeded := local_set.maximums_exceeded * local_set.ready_tasks;
2804 10663          local_set.ready_tasks := local_set.ready_tasks XOR (local_set.minimums_to_satisfy +
2804 10664              local_set.maximums_exceeded);
2804 10665
2804 10666          #unchecked_conversion (local_set, integer_dp_sets);
2804 10667
2804 10668          r := $real(integer_dp_sets);
2804 10669          priority.r := r;
2804 10670
2804 10671          ptlo := 0;
2804 10672        /find_next_task/
2804 10673          WHILE priority.dp_mask.dp <> 0 DO
285E 10674            ptlo := tmv$dct [priority.dp_mask.dp].queue_head;
285E 10675
285E 10676            WHILE ptlo <> 0 DO
2872 10677              ptle_p := ^tmv$ptl_p^ [ptlo];
2872 10678              cst_p^.ijl_ordinal := ptle_p^.ijl_ordinal;
2872 10679              jmp$get_ijle_p (cst_p^.ijl_ordinal, cst_p^.ijle_p);
2872 10680              cst_p^.ajlo := cst_p^.ijle_p^.ajl_ordinal;
2872 10681              cst_p^.jcb_p := #ADDRESS (1, cst_p^.ajlo + mtc$job_fixed_segment, 0);
2872 10682              xcb_p := #ADDRESS (1, mtc$job_fixed_segment + cst_p^.ajlo, ptle_p^.xcb_offset);
2872 10683
2872 10684              IF osv$cpus_logically_on > 1 THEN
28E6 10685                IF cst_p^.cst_index IN xcb_p^.processor_selections THEN
28FC 10686                  IF cst_p^.ijle_p^.multiprocessing_allowed THEN
```

```
2904 10687                    IF xcb_p^.last_lpid_for_task <> cst_p^.cst_index THEN
290C 10688                    { NOTE: The second parameter on the #purge_buffer command is just a dummy pointer.
290C 10689                      #PURGE_BUFFER (osc$purge_all_cache, xcb_p);
290E 10690                      #PURGE_BUFFER (osc$purge_all_page_seg_map, xcb_p);
2910 10691                    IFEND;
2910 10692                    EXIT /find_next_task/;
291A 10693                  ELSEIF cst_p^.ijle_p^.executing_task_count = 0 THEN
2922 10694                    IF cst_p^.jcb_p^.last_lpid_for_job <> cst_p^.cst_index THEN
292A 10695                      #PURGE_BUFFER (osc$purge_all_cache, xcb_p);
292C 10696                      #PURGE_BUFFER (osc$purge_all_page_seg_map, xcb_p);
292E 10697                    IFEND;
292E 10698                    EXIT /find_next_task/;
2936 10699                  IFEND;
2938 10700                IFEND;
293C 10701              ELSE
293C 10702                EXIT /find_next_task/;
2944 10703              IFEND;
2944 10704              ptlo := tmv$ptl_p^ [ptlo].ptl_thread;
2944 10705            WHILEND;
295A 10706
295A 10707 { No task of the priority selected could execute.  Remove the priority from the sets and select
295A 10708 { another priority.
295A 10709
295A 10710            local_set.minimums_to_satisfy := local_set.minimums_to_satisfy - $jmt$dispatching_priority_set
295A 10711                  [jmc$dp_conversion - priority.dp_mask.dp];
295A 10712            local_set.ready_tasks := local_set.ready_tasks - $jmt$dispatching_priority_set
295A 10713                  [jmc$dp_conversion - priority.dp_mask.dp];
295A 10714            local_set.maximums_exceeded := local_set.maximums_exceeded - $jmt$dispatching_priority_set
295A 10715                  [jmc$dp_conversion - priority.dp_mask.dp];
295A 10716
295A 10717            #unchecked_conversion (local_set, integer_dp_sets);
295A 10718
295A 10719            r := $real(integer_dp_sets);
295A 10720            priority.r := r;
295A 10721
295A 10722          WHILEND /find_next_task/;
29B2 10723          IF ptlo <> 0 THEN
29B6 10724            cst_p^.dispatching_priority_integer := integer_dp_sets;
29B6 10725            tmp$remove_task_from_dct (ptlo);
2B32 10726            cst_p^.ijle_p^.executing_task_count := cst_p^.ijle_p^.executing_task_count + 1;
2B54 10727          IFEND;
2B58 10728        IFEND;
2B5C 10729      IFEND;
2B5C 10730
2B5C 10731 {  Set up the CPU STATE TABLE.
2B5C 10732
2B5C 10733      IF ptlo = 0 THEN
2B60 10734        IF cst_p^.cpu_idle_statistics.idle_type = osc$not_idle THEN
2B68 10735          cst_p^.cpu_idle_statistics.idle_start_time := #FREE_RUNNING_CLOCK (0);
2B6E 10736          IF (tmv$swapin_in_progress <> 0) OR (tmv$io_wait_task_count <> 0) THEN
2B82 10737            cst_p^.cpu_idle_statistics.idle_type := osc$idle_with_io_active;
2B8A 10738          ELSE
2B8A 10739            cst_p^.cpu_idle_statistics.idle_type := osc$idle_no_io_active;
2B90 10740          IFEND;
2B90 10741        IFEND;
2B90 10742        cst_p^.dispatching_priority := 0;
```



```
2B90 10743        cst_p^.dispatching_priority_integer := 0;
2B90 10744        cst_p^.dual_state_prior_subpriority.dual_state_priority := 0;
2B90 10745        cst_p^.xcb_p := NIL;
2B90 10746        cst_p^.taskid.index := 0;
2B90 10747        IF (tmv$timed_wait_queue.head = 0) OR (tmv$dedicate_a_cpu_to_nos) THEN
2B94 10748          time := 0ffffffffffff(16);
2B9C 10749        ELSE
2B9C 10750          time := tmv$ptl_p^ [tmv$timed_wait_queue.head].end_of_wait_time;
2BCA 10751        IFEND;
2BCA 10752        tmv$time_to_call_dispatcher := time;
2BCA 10753        time := time - #free_running_clock (0);
2BD4 10754        IF time < 10 THEN
2BDC 10755          time := 10;
2BE2 10756        ELSEIF time > 50000 THEN
2BEE 10757          time := 50000;
2BF4 10758        IFEND;
2BF4 10759        cst_p^.max_cptime := time;
2BFC 10760
2BFC 10761      ELSE
2BFC 10762
2BFC 10763 { A new task has been selected.  Update the cpu_idle_statistics.
2BFC 10764
2BFC 10765        IF cst_p^.cpu_idle_statistics.idle_type <> osc$not_idle THEN
2C04 10766          idle_time := #FREE_RUNNING_CLOCK (0) - cst_p^.cpu_idle_statistics.idle_start_time;
2C0A 10767          IF cst_p^.cpu_idle_statistics.idle_type = osc$idle_with_io_active THEN
2C14 10768            cst_p^.cpu_idle_statistics.idle_io_active := cst_p^.cpu_idle_statistics.idle_io_active +
2C22 10769                  idle_time;
2C22 10770          ELSE
2C22 10771            cst_p^.cpu_idle_statistics.idle_no_io_active := cst_p^.cpu_idle_statistics.idle_no_io_active +
2C2C 10772                  idle_time;
2C2C 10773          IFEND;
2C2C 10774          update_dispatching_controls (idle_time, jmc$null_dispatching_priority);
2D18 10775          cst_p^.cpu_idle_statistics.idle_type := osc$not_idle;
2D18 10776          cst_p^.cpu_idle_statistics.idle_count := cst_p^.cpu_idle_statistics.idle_count + 1;
2D26 10777        IFEND;
2D26 10778
2D26 10779 { Set task status to executing and set up the rest of the CST fields.
2D26 10780
2D26 10781        ptle_p^.status := tmc$ts_executing;
2D26 10782
2D26 10783        cst_p^.taskid.index := ptlo;
2D26 10784        cst_p^.taskid.seqno := ptle_p^.sequence_number;
2D26 10785        cst_p^.xcb_p := xcb_p;
2D26 10786        cst_p^.max_cptime := xcb_p^.timeslice.minor;
2D26 10787        cst_p^.dispatching_priority := ptle_p^.dispatching_priority;
2D26 10788        cst_p^.dual_state_prior_subpriority := tmv$dual_state_dispatch_prior
2D26 10789                  [cst_p^.dispatching_priority];
2D26 10790        IF ptle_p^.monitor_flags <> $syt$monitor_flags [] THEN
2D60 10791          xcb_p^.xp.user_condition_register := xcb_p^.xp.user_condition_register +
2D60 10792                $ost$user_conditions [osc$free_flag];
2D60 10793          xcb_p^.monitor_flags := xcb_p^.monitor_flags + ptle_p^.monitor_flags;
2D60 10794          ptle_p^.monitor_flags := $syt$monitor_flags [];
2D7C 10795        IFEND;
2D7C 10796        IF ptle_p^.system_flags <> $tmt$system_flags [] THEN
2D84 10797          xcb_p^.system_flags := xcb_p^.system_flags + ptle_p^.system_flags;
2D84 10798          ptle_p^.system_flags := $tmt$system_flags [];
```

TMP$SWITCH_TASK

```
2D92 10799        IFEND;
2D92 10800
2D92 10801        IF (ptle_p^.ptl_flags.wait_inhibited = tmc$wi_wait_inhibited) THEN
2D9E 10802          xcb_p^.wait_inhibited := TRUE;
2DA2 10803        IFEND;
2DA2 10804        ptle_p^.ptl_flags.wait_inhibited := tmc$wi_null;
2DA2 10805
2DA2 10806 {  Set keypoint mask and keypoint enable flag in job exchange package.  Set the value for
2DA2 10807 {  monitor's keypoint mask.
2DA2 10808
2DA2 10809        #WRITE_REGISTER (osc$pr_set_keypoint_enable, $integer (xcb_p^.keypoint_register_enable));
2DB8 10810
2DB8 10811 {  Update the next time to ready a task from the timed wait queue.
2DB8 10812
2DB8 10813        IF (tmv$timed_wait_queue.head = 0) THEN
2DC0 10814          tmv$time_to_call_dispatcher := 0ffffffffffff(16);
2DCC 10815        ELSE
2DCC 10816          tmv$time_to_call_dispatcher := tmv$ptl_p^ [tmv$timed_wait_queue.head].end_of_wait_time;
2DDC 10817        IFEND;
2DDC 10818
2DDC 10819      IFEND;
2DDC 10820      tmp$clear_lock (tmv$ptl_lock);
2E14 10821
2E14 10822      #KEYPOINT (osk$mtr, osk$monitor_multiplier * ptlo, tmk$switch_task + osk$m);
2E1C 10823
2E1C 10824    PROCEND tmp$switch_task;
   0 10825
```


TMP$SET_UP_DEBUG_REGISTERS

```
   0 10827
   0 10828 { PURPOSE:
   0 10829 {   This procedure is called during the first swapin for job recovery to straighten out the
   0 10830 {   debug list and mask in the exchange package of recovering jobs.
   0 10831
   0 10832   PROCEDURE [XDCL] tmp$set_up_debug_registers
   0 10833     (     ptlo: ost$task_index;
   0 10834           ijle_p: ^jmt$initiated_job_list_entry;
   0 10835           xcb_p: ^ost$execution_control_block);
   0 10836
   0 10837     VAR
   0 10838       null_debug_mask: [STATIC, READ] ost$debug_mask := [FALSE, FALSE, [REP 5 of FALSE]];
   0 10839
   0 10840     IF (xcb_p^.xp.debug_list_pointer <> NIL) AND (#RING (xcb_p^.xp.debug_list_pointer) = 1) THEN
  20 10841       xcb_p^.xp.debug_list_pointer := NIL;
  20 10842       xcb_p^.xp.debug_mask_register := null_debug_mask;
  20 10843
  20 10844       xcb_p^.xp.debug_index := 0;
  20 10845       IF osc$debug IN xcb_p^.xp.user_mask THEN
  56 10846         xcb_p^.xp.user_mask := xcb_p^.xp.user_mask - $ost$user_conditions [osc$debug];
  56 10847         xcb_p^.monitor_flags := xcb_p^.monitor_flags + $syt$monitor_flags [syc$mf_system_debugger];
  56 10848         IF tmv$ptl_p^ [ptlo].ptl_flags.wait_inhibited <> tmc$wi_wait_selected_r3 THEN
  88 10849           xcb_p^.xp.user_condition_register := xcb_p^.xp.user_condition_register +
  96 10850                 $ost$user_conditions [osc$free_flag];
  96 10851         IFEND;
  96 10852       IFEND;
  96 10853     IFEND;
  96 10854   PROCEND tmp$set_up_debug_registers;
   0 10855
```

TMP$UPDATE_DEBUG_REGISTERS

```
        o 10857
        o 10858 { PURPOSE:
        o 10859 {   This procedure sets the debugger flag for all tasks.  The next time a task executes, it will trap to
        o 10860 {   reset the debug list and mask in its exchange package.  This procedure is called whenever a breakpoint
        o 10861 {   is selected, set, removed, or modified during a debugger session.
        o 10862
        o 10863   PROCEDURE tmp$update_debug_registers;
        o 10864
        o 10865     VAR
        o 10866       ptlo: ost$task_index;
        o 10867
        o 10868     FOR ptlo := 1 TO UPPERBOUND (tmv$ptl_p^) DO
      1A 10869       IF tmv$ptl_p^ [ptlo].status <> tmc$ts_null THEN
      2A 10870         tmv$ptl_p^ [ptlo].monitor_flags := tmv$ptl_p^ [ptlo].monitor_flags +
      38 10871                 $syt$monitor_flags [syc$mf_system_debugger];
      38 10872       IFEND;
      38 10873     FOREND;
      3C 10874
      3C 10875   PROCEND tmp$update_debug_registers;
         10876
```


UPDATE_TIMED_WAIT_QUEUE

```
         10878
         10879   PROCEDURE [INLINE] update_timed_wait_queue;
         10880
         10881 {  This procedure is called during task switch to remove tasks from the timed wait
         10882 {  queue if the wait time has expired.
         10883 {  NOTE: this routine is called with the PTL lock set.
         10884
         10885     VAR
         10886       attempt_preselection: boolean,
         10887       free_running_clock: integer,
         10888       ijle_p: ^jmt$initiated_job_list_entry,
         10889       xcb_p: ^ost$execution_control_block,
         10890       ptlo: ost$task_index,
         10891       ptle_p: ^tmt$primary_task_list_entry;
         10892
         10893     free_running_clock := #FREE_RUNNING_CLOCK (0);
         10894
         10895 {UPDATE THE TIMED_WAIT_QUEUE
         10896     WHILE (tmv$ptl_p^ [tmv$timed_wait_queue.head].end_of_wait_time <= free_running_clock) AND
         10897         (tmv$timed_wait_queue.head <> 0) DO
         10898       ptlo := tmv$timed_wait_queue.head;
         10899       ptle_p := ^tmv$ptl_p^ [ptlo];
         10900       tmv$timed_wait_queue.head := ptle_p^.queue_link.head;
         10901       tmv$ptl_p^ [ptle_p^.queue_link.head].queue_link.tail := 0;
         10902       ptle_p^.queue_link.head := 0;
         10903       IF (ptle_p^.status <= tmc$ts_last_status_in_dct) OR (ptle_p^.status > tmc$ts_last_status_in_wait_q) THEN
         10904         mtp$error_stop ('TM36 - task not in wait queue');
         10905       IFEND;
         10906       jmp$get_ijle_p (ptle_p^.ijl_ordinal, ijle_p);
         10907       IF ptle_p^.idle_status = tmc$is_idled THEN
         10908         ptle_p^.idle_status := tmc$is_idled_sched_notified;
         10909         IF ptle_p^.status = tmc$ts_timeout_reqexp_longvlong THEN
         10910           ijle_p^.statistics.tasks_not_in_long_wait := ijle_p^.statistics.tasks_not_in_long_wait + 1;
         10911         IFEND;
         10912         ptle_p^.status := tmc$ts_ready_but_swapped;
         10913         jmp$ready_task_in_swapped_job (ptle_p^.ijl_ordinal, ijle_p);
         10914         ijle_p^.long_wait_aging_complete := FALSE;
         10915       ELSE
         10916         IF ptle_p^.status = tmc$ts_timeout_reqexp_longvlong THEN
         10917           ijle_p^.statistics.tasks_not_in_long_wait := ijle_p^.statistics.tasks_not_in_long_wait + 1;
         10918           jmv$ajl_p^ [ijle_p^.ajl_ordinal].job_is_good_swap_candidate := FALSE;
         10919         IFEND;
         10920         ptle_p^.status := tmc$ts_ready;
         10921         xcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, ptle_p^.xcb_offset);
         10922         xcb_p^.timeslice.minor := 0;
         10923         xcb_p^.timeslice.major := 0;
         10924         attempt_preselection := (ijle_p^.multiprocessing_allowed) OR (ijle_p^.executing_task_count = 0);
         10925         tmp$dct_ready_task (xcb_p, ijle_p, ptlo, attempt_preselection);
         10926       IFEND;
         10927       ijle_p^.statistics.ready_task_count := ijle_p^.statistics.ready_task_count + 1;
         10928     WHILEND;
         10929
         10930 { Update the queue's tail.
         10931
         10932     IF tmv$timed_wait_queue.head = 0 THEN
         10933       tmv$timed_wait_queue.tail := 0;
```

UPDATE_TIMED_WAIT_QUEUE

```
10934      IFEND;
10935
10936    PROCEND update_timed_wait_queue;
10937
```


TMP$JOB_RECOVERY_REQUESTS

```
10939 { monitor requests for job recovery
10940
10941    TYPE
10942      syt$rb_job_recovery = record
10943        reqcode: ALIGNED [0 MOD 8] syt$monitor_request_code,
10944        status: syt$monitor_status,
10945        ijlo: jmt$ijl_ordinal,
10946        case subreq: syt$job_recovery_subreq of
10947        = syc$recover_ptl =
10948          count: integer,
10949          task_list_p: ^array [1 .. * ] of syt$ptl_recovery_info,
10950        casend,
10951      recend,
10952
10953      syt$ptl_recovery_info = record
10954        xcb_offset: 0 .. 7fffffff(16),
10955        dispatching_priority: jmt$dispatching_priority,
10956        gtid: ost$global_task_id,
10957      recend,
10958
10959      syt$job_recovery_subreq = (syc$recover_ptl, syc$dummy_job_recovery);
10960
 o 10963
 o 10964    PROCEDURE [XDCL] tmp$job_recovery_requests
 o 10965      (VAR rb: syt$rb_job_recovery;
 o 10966           cst_p: ^ost$cpu_state_table);
 o 10967
 o 10968      VAR
 o 10969        j,
 o 10970        i: integer,
 o 10971        gtid: ost$global_task_id,
 o 10972        jmtr_ptlo,
 o 10973        ptlo: ost$task_index,
 o 10974        ijle_p: ^jmt$initiated_job_list_entry,
 o 10975        ptl_p: ^tmt$primary_task_list_entry,
 o 10976        service_class: jmt$service_class_index;
 o 10977
 o 10978      rb.status.normal := TRUE;
 4 10979      CASE rb.subreq OF
16 10980      = syc$recover_ptl =
16 10981        tmp$set_lock (tmv$ptl_lock);
52 10982
52 10983        FOR i := 1 TO rb.count DO
60 10984          IF tmv$ptl_p^ [rb.task_list_p^ [i].gtid.index].dispatching_priority <> jmc$null_dispatching_priority
7E 10985              THEN
7E 10986            rb.status.normal := FALSE;
7E 10987            rb.status.condition := tme$ptl_full;
7E 10988            tmp$clear_lock (tmv$ptl_lock);
C2 10989            RETURN;
C4 10990          IFEND;
C4 10991        FOREND;
C8 10992
C8 10993        jmtr_ptlo := jmv$ijl_p.block_p^ [rb.ijlo.block_number].index_p^ [rb.ijlo.block_index].
C8 10994                     job_monitor_taskid.index;
C8 10995        FOR i := 1 TO rb.count DO
FE 10996          ptlo := rb.task_list_p^ [i].gtid.index;
```

```
 FE 10997          remove_task_from_free_queue (ptlo);
1D6 10998          ptl_p := ^tmv$ptl_p^ [ptlo];
1D6 10999          ptl_p^.ptl_thread := 0;
1D6 11000          ptl_p^.status := tmc$ts_timeout_reqexp_inflong;
1D6 11001          ptl_p^.ijl_ordinal := rb.ijlo;
1D6 11002          ptl_p^.xcb_offset := rb.task_list_p^ [i].xcb_offset;
1D6 11003          ptl_p^.dispatching_priority := rb.task_list_p^ [i].dispatching_priority;
1D6 11004          ptl_p^.monitor_flags := $syt$monitor_flags [];
1D6 11005          ptl_p^.system_flags := $tmt$system_flags [];
1D6 11006          ptl_p^.sequence_number := rb.task_list_p^ [i].gtid.seqno;
1D6 11007          ptl_p^.idle_status := tmc$is_idled;
1D6 11008          ptl_p^.queue_link.head := 0;
1D6 11009          ptl_p^.queue_link.tail := 0;
1D6 11010          ptl_p^.end_of_wait_time := 0;
1D6 11011
1D6 11012 { Emulate insert_ijl here - note special case for first task (jmtr).
1D6 11013
1D6 11014          IF ptlo <> jmtr_ptlo THEN
23E 11015            tmv$ptl_p^ [ptlo].ijl_thread := tmv$ptl_p^ [jmtr_ptlo].ijl_thread;
23E 11016            tmv$ptl_p^ [jmtr_ptlo].ijl_thread := ptlo;
24E 11017          IFEND;
24E 11018          gtid.index := ptlo;
24E 11019          gtid.seqno := ptl_p^.sequence_number;
24E 11020          tmp$set_task_ready (gtid, 0 {readying_task_priority}, tmc$rc_ready_conditional_wi);
276 11021        FOREND;
280 11022        tmp$clear_lock (tmv$ptl_lock);
2BA 11023
2BA 11024 { Get an ijl pointer - we need to reset some ijl fields and straighten out some counts.
2BA 11025
2BA 11026        jmp$get_ijle_p (rb.ijlo, ijle_p);
2BA 11027
2BA 11028 { Set the swap status of the job to swapout complete; clear the swap queue link fields and
2BA 11029 { relink the job into the swapped_out swap queue.
2BA 11030
2BA 11031        ijle_p^.swap_status := jmc$iss_swapout_complete;
2BA 11032        ijle_p^.swap_queue_link.queue_id := jsc$isqi_null;
2BA 11033        ijle_p^.swap_queue_link.backward_link := jmv$null_ijl_ordinal;
2BA 11034        ijle_p^.swap_queue_link.forward_link := jmv$null_ijl_ordinal;
2BA 11035        jsp$relink_swap_queue (rb.ijlo, ijle_p, jsc$isqi_swapped_out);
31C 11036
31C 11037 { Increment the total task count by the number of tasks in the recovered job.
31C 11038
31C 11039        tmv$total_task_count := tmv$total_task_count + rb.count;
31C 11040
31C 11041 { Increment the scheduler initiated job count so that scheduler knows how many jobs of this class
31C 11042 { there are; this count is normally incremented in jmp$initiate_job_from_scheduler, but for job
31C 11043 { recovery we'll do it here.
31C 11044
31C 11045        service_class := ijle_p^.job_scheduler_data.service_class;
31C 11046        jmv$job_counts.service_class_counts [service_class].scheduler_initiated_jobs := jmv$job_counts.
31C 11047              service_class_counts [service_class].scheduler_initiated_jobs + 1;
31C 11048
31C 11049 { Increment swapped job count so that scheduler knows how many swapped jobs there are.
31C 11050
31C 11051        jmp$increment_swapped_job_count (ijle_p);
354 11052
```



```
354 11053      ELSE
354 11054
354 11055 { unsupported subrequest
354 11056
354 11057      CASEND;
354 11058    PROCEND tmp$job_recovery_requests;
  0 11059
```

[XDCL] tmp$free_unrecovered_tasks

```
        0 11061
        0 11062 { PURPOSE:
        0 11063 {   This procedure frees the PTL entries for the tasks of a job that is being terminated
        0 11064 {   because it cannot be recoverd.  The total task count is decremented; it was incremented
        0 11065 {   earlier during job recovery when the PTL entries were made.
        0 11066 {   The caller of this procedure MUST set the PTL lock.
        0 11067
        0 11068   PROCEDURE [XDCL] tmp$free_unrecovered_tasks
        0 11069     (     ijle_p: ^jmt$initiated_job_list_entry);
        0 11070
        0 11071     VAR
        0 11072       next_index: ost$task_index,
        0 11073       task_index: ost$task_index;
        0 11074
        0 11075     task_index := ijle_p^.job_monitor_taskid.index;
        4 11076     WHILE task_index <> 0 DO
       16 11077       next_index := tmv$ptl_p^ [task_index].ijl_thread;
       16 11078       IF tmv$ptl_p^ [task_index].ptl_thread <> 0 THEN
       2A 11079         mtp$error_stop ('Unrecovered task--PTL thread not zero');
       4A 11080       IFEND;
       4A 11081
       4A 11082 { The procedure free_ptl does not clear the ijl thread. Not clearing the field
       4A 11083 { causes a potential problem for the next task using the ptl entry.
       4A 11084
       4A 11085       tmv$ptl_p^ [task_index].ijl_thread := 0;
       4A 11086       free_ptl (task_index);
       84 11087       tmv$total_task_count := tmv$total_task_count - 1;
       84 11088       task_index := next_index;
       84 11089     WHILEND;
       9A 11090
       9A 11091   PROCEND tmp$free_unrecovered_tasks;
        0 11092
```


TMP$SWITCH_TASK_FROM_FAILING_CP

```
        0 11094
        0 11095   PROCEDURE [XDCL] tmp$switch_task_from_failing_cp
        0 11096     (VAR cst_p: ^ost$cpu_state_table);
        0 11097
        0 11098     VAR
        0 11099       next_task_ijle_p: ^jmt$initiated_job_list_entry,
        0 11100       next_task_xcb_p: ^ost$execution_control_block,
        0 11101       ptle_p: ^tmt$primary_task_list_entry,
        0 11102       ptlo: ost$task_index,
        0 11103       xcb_p: ^ost$execution_control_block;
        0 11104
        0 11105     xcb_p := cst_p^.xcb_p;
        4 11106
        4 11107     tmp$set_lock (tmv$ptl_lock);
       4E 11108
       4E 11109     IF xcb_p <> NIL THEN
       58 11110       xcb_p^.timeslice := jmv$service_classes [cst_p^.ijle_p^.job_scheduler_data.service_class]^.attributes.
       58 11111             dispatching_control [cst_p^.ijle_p^.dispatching_control.dispatching_control_index].
       58 11112             dispatching_timeslice;
       58 11113       xcb_p^.last_lpid_for_task := cst_p^.cst_index;
       58 11114       cst_p^.jcb_p^.last_lpid_for_job := cst_p^.cst_index;
       58 11115       cst_p^.jcb_p^.last_execution_time := #FREE_RUNNING_CLOCK (0);
       9C 11116       ptlo := cst_p^.taskid.index;
       9C 11117       ptle_p := ^tmv$ptl_p^ [ptlo];
       9C 11118       cst_p^.ijle_p^.executing_task_count := cst_p^.ijle_p^.executing_task_count - 1;
       9C 11119       IF ptle_p^.new_task_status > tmc$ts_last_status_in_dct THEN
       C6 11120         ptle_p^.status := ptle_p^.new_task_status;
       C6 11121         ptle_p^.new_task_status := tmc$ts_null;
       C6 11122         cst_p^.ijle_p^.statistics.ready_task_count := cst_p^.ijle_p^.statistics.ready_task_count - 1;
       C6 11123         IF ptle_p^.status <= tmc$ts_last_status_in_wait_q THEN
       DE 11124           tmp$insert_timed_wait_queue (ptlo);
      1A6 11125         IFEND;
      1A6 11126         IF (ptle_p^.status = tmc$ts_timeout_reqexp_longvlong) OR (ptle_p^.status =
      1BC 11127               tmc$ts_timed_wait_not_queued) OR (ptle_p^.status = tmc$ts_timeout_reqexp_infvlong) THEN
      1BC 11128           cst_p^.ijle_p^.statistics.tasks_not_in_long_wait := cst_p^.ijle_p^.statistics.
      1BC 11129                 tasks_not_in_long_wait - 1;
      1BC 11130           IF (cst_p^.ijle_p^.statistics.tasks_not_in_long_wait = 0) AND
      1E4 11131                 (cst_p^.ijle_p^.entry_status = jmc$ies_job_in_memory) AND
      1E4 11132                 (ptle_p^.idle_status = tmc$is_not_idled) THEN
      1E4 11133             cst_p^.xcb_p := NIL;
      1E4 11134             tmp$set_swapout_candidate (cst_p^.ajlo);
      1FC 11135           IFEND;
      1FC 11136         IFEND;
      200 11137       ELSE
      200 11138         ptle_p^.status := tmc$ts_ready;
      200 11139         tmp$dct_ready_task (xcb_p, cst_p^.ijle_p, ptlo, TRUE);
      AEA 11140       IFEND;
      AEA 11141     IFEND;
      AEA 11142
      AEA 11143 { If a task has been pre-selected to run on this CPU, that task needs to be inserted in the DCT.
      AEA 11144
      AEA 11145     IF cst_p^.next_ptlo_to_dispatch <> 0 THEN
      AF6 11146       jmp$get_ijle_p (tmv$ptl_p^ [cst_p^.next_ptlo_to_dispatch].ijl_ordinal, next_task_ijle_p);
      AF6 11147       next_task_ijle_p^.executing_task_count := next_task_ijle_p^.executing_task_count - 1;
      AF6 11148       tmv$ptl_p^ [cst_p^.next_ptlo_to_dispatch].status := tmc$ts_ready;
      AF6 11149       tmp$get_xcb_p_from_ptlo (cst_p^.next_ptlo_to_dispatch, next_task_ijle_p^.ajl_ordinal,
```

TMP$SWITCH_TASK_FROM_FAILING_CP

```
  AF6 11150              next_task_xcb_p);
  AF6 11151          tmp$dct_ready_task (next_task_xcb_p, next_task_ijle_p, cst_p^.next_ptlo_to_dispatch, TRUE);
 1402 11152          cst_p^.next_ptlo_to_dispatch := 0;
 140A 11153        IFEND;
 140A 11154
 140A 11155        tmp$clear_lock (tmv$ptl_lock);
 1442 11156
 1442 11157     PROCEND tmp$switch_task_from_failing_cp;
    0 11159 MODEND tmm$dispatcher;

**** 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	7866	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	7866	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	7866	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	7866	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	7898	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8227	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	8739	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9406	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9406	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9479	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9479	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9479	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9479	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9479	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9623	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9623	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9623	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9623	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9623	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9647	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	9647	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10201	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10201	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10201	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10201	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10201	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10466	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10518	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10550	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10550	Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY

**** 45 warning diagnostics

IDENTIFIER	DEFINED ON LINE	REFERENCES							
a0_dynamic_space_pointer	3104	7593							
accumulated_job_cptime	1161	8738/M	8738	8739	8739	8739	8739	9688	9690
		9703	9710	9929	9961/M	9961	10465	10466	10466
		10466	10466						
accumulated_monitor_cptime	1162	8732/M	8739	8739	8739	8739	9686	9690	9700
		9708	10466	10466	10466	10466			
activity	8763	8765							
activity	8772	8787	8791	8802					
ajl_ordinal	1267	5252	5293	5493	5698	5716	7054	7054	7056
		7056	7058	7106	7149	7218	7258	7320	7459
		7463	7505	7509	7539	7543	7577	7581	7598
		7668	7673	7685	7698	7702	7716	7810	7866
		7866	7866	7866	7866	7896/S	7898	7898	7898
		7898	7898	7978	7979	7979	7979	7979	7979
		8170	8172	8172	8172	8172	8172	8218	8222
		8226/S	8227	8227	8227	8227	8227	8232	8499
		8499	8499	8499	8499	8630	8630	8630	8630
		8630	9406	9406	9406	9406	9406	9479	9479
		9479	9479	9479	9623	9623	9623	9623	9623
		9647	9647	9647	9647	9647	10201	10201	10201
		10201	10201	10515/P	10519	10519	10519	10519	10519
		10550/S	10550	10550	10550	10550	10550	10550	10579
		10579	10579	10579	10579	10637	10680	10918/S	10921
		10925	10925	10925	10925	10925	11139	11139	11139
		11139	11139	11149/P	11151	11151	11151	11151	11151
ajl_ordinal	4828	7660/M	7666/S	7673/M	7702/M	7721			
ajl_ordinal	5676	5698/M	5698	5698/S	5698/S	5698			
ajl_ordinal	5713	5716/M	5717	5721/S	5721/S	5722			
ajl_ordinal	7416	7459/M	7459	7459/S	7459/S	7459			
ajl_ordinal	7472	7505/M	7505	7505/S	7505/S	7505			
ajl_ordinal	7517	7539/M	7539	7539/S	7539/S	7539			
ajl_ordinal	7562	7577/M	7577	7577/S	7577/S	7577			
ajl_ordinal	7609	7612							
ajl_ordinal	7618	7668/M	7668	7668/S	7668/S	7668	7698/M	7698	7698/S
		7698/S	7698						
ajl_ordinal	8416	8428/P	8433						
ajl_ordinal	9217	9255							
ajl_ordinal	9218	9227	9231/S	9255/P	9268/P				
ajl_ordinal	9279	9287	9291/S	9302	9310/P				
ajl_ordinal	9323	9331/S	9333/P	9335/P					
ajl_ordinal	9350	9366/S	9368	9372/S	9385				
ajl_ordinal	9447	9458/S	9461/S	9471					
ajl_ordinal	10408	10516							
ajl_ordinal	11095	11150							
ajlo	1153	8425	8606	8754	9140	9887	10530	10571/P	10637/M
		10638	10639	10680/M	10681	10682	11134/P		
ajlo	5491	5493/M	5494/S	5497/S	5497/S				
ajlo	5676	5698/P	5698/M						
ajlo	5691	5698/P							
ajlo	5710	5719/P	5722/M						
ajlo	7416	7459/P							
ajlo	7416	7459/P	7459/M						
ajlo	7472	7505/P							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ajlo	7472	7505/P	7505/M						
ajlo	7517	7539/P							
ajlo	7517	7539/P	7539/M						
ajlo	7562	7577/P							
ajlo	7562	7577/P	7577/M						
ajlo	7562	7598/M	7598/S	7598/S	7598/S				
ajlo	7618	7668/P	7698/P						
ajlo	7618	7668/P	7668/M	7698/P	7698/M				
ajlo	7618	7685/M	7685/S	7685/S	7685/S	7716/M	7716/S	7716/S	7716/S
ajo	4989	8433/M	8437	8438/S	8450/M	8459	8475	8493/P	
amc$file_byte_limit	3923	3926	3928						
amc$max_file_id_ordinal	6030	6037							
amt$file_byte_address	3926	3887							
amt$file_id_ordinal	6037	6034							
amt$file_id_sequence	6038	6035							
amt$file_identifier	6033	5997							
amt$file_limit	3928	3891							
asid	2700	8427/P							
asti	8206	8230	8231/S	8232/S					
attempt_preselection	6867	6938							
attempt_preselection	7766	7866	7898						
attempt_preselection	7795	7865/M	7866/P	7897/M	7898/P				
attempt_preselection	7932	7979							
attempt_preselection	7950	7977/M	7979/P						
attempt_preselection	8110	8172							
attempt_preselection	8128	8171/M	8172/P						
attempt_preselection	8182	8227							
attempt_preselection	8381	8499							
attempt_preselection	8400	8498/M	8499/P						
attempt_preselection	8581	8630							
attempt_preselection	8598	8627/M	8630/P						
attempt_preselection	9349	9406							
attempt_preselection	9355	9404/M	9406/P						
attempt_preselection	9446	9479							
attempt_preselection	9450	9478/M	9479/P						
attempt_preselection	9584	9623	9647						
attempt_preselection	9589	9622/M	9623/P	9646/M	9647/P				
attempt_preselection	10157	10201							
attempt_preselection	10163	10200/M	10201/P						
attempt_preselection	10408	10519	10550	10579					
attempt_preselection	10408	10550/M	10550/P						
attempt_preselection	10423	10577/M	10579/P						
attempt_preselection	10879	10925							
attempt_preselection	10886	10924/M	10925/P						
attempt_preselection	11095	11139	11151						
attributes	6263	6884	7866	7898	7979	8172	8227	8496	8499
		8624	8630	9198	9406	9479	9623	9647	10045
		10056	10193	10201	10277	10330	10337	10373	10497
		10519	10550	10579	10925	11110	11139	11151	
avc$monitor_statistics_flag	3605	8739/P	9745/P	10466/P					
b	5286	5295	5295						
b	5330	5340	5341						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES			
b	6424	6431	6432		
b	6464	6490	6490		
b	7416	7450	7450		
b	7472	7501	7501		
b	7517	7529	7529		
b	7562	7573	7573		
b	7618	7653	7653		
b	7730	7748	7748		
b	7766	7806	7806		
b	7932	7956	7956		
b	8011	8017	8017		
b	8044	8070	8070		
b	8110	8134	8134		
b	8182	8222	8222	8232	8232
b	8259	8265	8265		
b	8276	8282	8282		
b	8381	8444	8444		
b	8513	8532	8532		
b	8581	8607	8607		
b	8682	8727	8727		
b	8770	8783	8783		
b	8875	8897	8897		
b	8966	9004	9004		
b	9057	9100	9100		
b	9217	9247	9247		
b	9322	9329	9329		
b	9446	9469	9469		
b	9491	9504	9504		
b	9584	9607	9607		
b	9936	9951	9951		
b	10157	10170	10170		
b	10224	10233	10233		
b	10408	10461	10461	10593	10593
b	10964	10981	10981		
b	11095	11107	11107		
backward_link	2031	11033/M			
bc	6425	6428/M	6429	6433	
bc	6464	6490/M	6490	6490	
bc	7416	7450/M	7450	7450	
bc	7472	7501/M	7501	7501	
bc	7517	7529/M	7529	7529	
bc	7562	7573/M	7573	7573	
bc	7618	7653/M	7653	7653	
bc	7730	7748/M	7748	7748	
bc	7766	7806/M	7806	7806	
bc	7932	7956/M	7956	7956	
bc	8011	8017/M	8017	8017	
bc	8044	8070/M	8070	8070	
bc	8110	8134/M	8134	8134	
bc	8259	8265/M	8265	8265	
bc	8276	8282/M	8282	8282	
bc	8381	8444/M	8444	8444	
bc	8513	8532/M	8532	8532	

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
bc	8581	8607/M	8607	8607					
bc	8682	8727/M	8727	8727					
bc	8770	8783/M	8783	8783					
bc	8875	8897/M	8897	8897					
bc	8966	9004/M	9004	9004					
bc	9057	9100/M	9100	9100					
bc	9217	9247/M	9247	9247					
bc	9322	9329/M	9329	9329					
bc	9446	9469/M	9469	9469					
bc	9491	9504/M	9504	9504					
bc	9584	9607/M	9607	9607					
bc	9936	9951/M	9951	9951					
bc	10157	10170/M	10170	10170					
bc	10224	10233/M	10233	10233					
bc	10408	10461/M	10461	10461	10593/M	10593	10593		
bc	10964	10981/M	10981	10981					
bc	11095	11107/M	11107	11107					
block_index	1243	5389/S	6612/S	6966/S	7458/S	7503/S	7537/S	7575/S	7665/S
		7688	7696/M	7809/S	7866/S	7898/S	7959/S	7979/S	8140/S
		8172/S	8227/S	8439/S	8445/S	8499/S	8608/S	8630/S	8837/S
		9196/S	9406/S	9458/S	9479/S	9596/S	9623/S	9647/S	9969/S
		10201/S	10512/S	10519/S	10550/S	10550/S	10579/S	10636/S	10679/S
		10906/S	10925/S	10993/S	11026/S	11139/S	11146/S	11151/S	
block_number	1242	5389/S	6612/S	6966/S	7458/S	7503/S	7537/S	7575/S	7665/S
		7691	7697/M	7809/S	7866/S	7898/S	7959/S	7979/S	8140/S
		8172/S	8227/S	8439/S	8445/S	8499/S	8608/S	8630/S	8837/S
		9196/S	9406/S	9458/S	9479/S	9596/S	9623/S	9647/S	9969/S
		10201/S	10512/S	10519/S	10550/S	10550/S	10579/S	10636/S	10679/S
		10906/S	10925/S	10993/S	11026/S	11139/S	11146/S	11151/S	
block_p	4063	5389	6612	6966	7458	7503	7537	7575	7665
		7692	7694	7809	7866	7898	7959	7979	8140
		8172	8227	8439	8445	8499	8608	8630	8837
		9196	9406	9458	9479	9596	9623	9647	9969
		10201	10512	10519	10550	10550	10579	10636	10679
		10906	10925	10993	11026	11139	11146	11151	
calculate_circular_service	10279	10279	10288	10291					
calculate_service_used	10262	9996	10063	10297					
call_dispatcher	3724	6990/M	7758/M	7866/M	7898/M	7979/M	8024/M	8092/M	8172/M
		8227/M	8499/M	8559/M	8630/M	8746/M	8815/M	8825/M	8933/M
		9049/M	9174/M	9406/M	9479/M	9623/M	9647/M	10084/M	10201/M
		10519/M	10550/M	10579/M	10925/M	11139/M	11151/M		
circular_service	10267	10276/M	10285/M	10285	10292	10293			
clear	4741	6453/M	6494/M	6502/M	7454/M	7466/M	7512/M	7547/M	7599/M
		7725/M	7757/M	7927/M	7983/M	8039/M	8104/M	8176/M	8270/M
		8292/M	8451/M	8508/M	8553/M	8610/M	8677/M	8756/M	8870/M
		8961/M	9010/M	9028/M	9048/M	9104/M	9211/M	9250/M	9257/M
		9271/M	9339/M	9486/M	9539/M	9573/M	9631/M	9655/M	9975/M
		10137/M	10141/M	10210/M	10246/M	10523/M	10589/M	10820/M	10988/M
		11022/M	11155/M						
cmc$on	1190	10481							
cmt$element_state	1190	1149	1150	1180					
condition	275	5651/M	6496/M	6534/M	6804/M	7452/M	7461/M	8445/M	8608/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8704/M	9102/M	9373/M	9412/M	9433/M	9543/M	9577/M	9976/M
		10987/M							
ondition	5646	5651							
ondition	6464	6496							
ondition	6505	6534							
ondition	6567	6604							
ondition	7416	7452	7461						
ondition	8381	8445							
ondition	8581	8608							
ondition	8682	8704							
ondition	9057	9102							
ondition	9349	9373	9412	9433					
ondition	9491	9543	9577						
ondition	9936	9976							
ontrols	4790	8739	9694	9762	10466	10774			
ontrols_defined	4786	8739	9694	9760	10466	10774			
ount	4738	6435/M	6435	6450	6451/M	6451	6490/M	6490	6494
		6494/M	6494	6502	6502/M	6502	7450/M	7450	7454
		7454/M	7454	7466	7466/M	7466	7501/M	7501	7512
		7512/M	7512	7529/M	7529	7547	7547/M	7547	7573/M
		7573	7599	7599/M	7599	7653/M	7653	7725	7725/M
		7725	7748/M	7748	7757	7757/M	7757	7806/M	7806
		7927	7927/M	7927	7956/M	7956	7983	7983/M	7983
		8017/M	8017	8039	8039/M	8039	8070/M	8070	8104
		8104/M	8104	8134/M	8134	8176	8176/M	8176	8265/M
		8265	8270	8270/M	8270	8282/M	8282	8292	8292/M
		8292	8444/M	8444	8451	8451/M	8451	8508	8508/M
		8508	8532/M	8532	8553	8553/M	8553	8607/M	8607
		8610	8610/M	8610	8677	8677/M	8677	8727/M	8727
		8756	8756/M	8756	8783/M	8783	8870	8870/M	8870
		8897/M	8897	8961	8961/M	8961	9004/M	9004	9010
		9010/M	9010	9028	9028/M	9028	9048	9048/M	9048
		9100/M	9100	9104	9104/M	9104	9211	9211/M	9211
		9247/M	9247	9250	9250/M	9250	9257	9257/M	9257
		9271	9271/M	9271	9329/M	9329	9339	9339/M	9339
		9469/M	9469	9486	9486/M	9486	9504/M	9504	9539
		9539/M	9539	9573	9573/M	9573	9607/M	9607	9631
		9631/M	9631	9655	9655/M	9655	9951/M	9951	9975
		9975/M	9975	10137	10137/M	10137	10141	10141/M	10141
		10170/M	10170	10210	10210/M	10210	10233/M	10233	10246
		10246/M	10246	10461/M	10461	10523	10523/M	10523	10589
		10589/M	10589	10593/M	10593	10820	10820/M	10820	10981/M
		10981	10988	10988/M	10988	11022	11022/M	11022	11107/M
		11107	11155	11155/M	11155				
ount	10948	10983	10995	11039					
p_service_at_class_switch	1454	10275							
p_time	1642	8739/M	8739	8739/M	8739	8739	8739	9685/M	9685
		9687/M	9687	9742	9743	10274	10275	10466/M	10466
		10466/M	10466	10466	10466				
p_time	2541	8739/M	8739	8739/M	8739	9707/M	9707	9709/M	9709
		9921	9930	10466/M	10466	10466/M	10466		
ptime_signal_last_sent	2251	8739	8739/M	9744	9746/M	10466	10466/M		
pu_idle_statistics	1174	10734	10735/M	10737/M	10739/M	10765	10766	10767	10768/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10768	10771/M	10771	10775/M	10776/M	10776		
pu_state	1156	10480	10587						
st_index	1148	6993	7866	7898	7979	8172	8227	8499	8630
		8744	9406	9479	9623	9647	10201	10469	10470
		10519	10550	10579	10642	10648	10685	10687	10694
		10925	11113	11114	11139	11151			
st_index	6870	6942/M	6947/S	6948	6949	6952	6953/S	6955/M	6955
		6956							
st_index	7766	7866/M	7866/S	7866	7866	7866	7866/S	7866/M	7866
		7866	7898/M	7898/S	7898	7898	7898	7898/S	7898/M
		7898	7898						
st_index	7932	7979/M	7979/S	7979	7979	7979	7979/S	7979/M	7979
		7979							
st_index	8110	8172/M	8172/S	8172	8172	8172	8172/S	8172/M	8172
		8172							
st_index	8182	8227/M	8227/S	8227	8227	8227	8227/S	8227/M	8227
		8227							
st_index	8381	8499/M	8499/S	8499	8499	8499	8499/S	8499/M	8499
		8499							
st_index	8581	8630/M	8630/S	8630	8630	8630	8630/S	8630/M	8630
		8630							
st_index	9349	9406/M	9406/S	9406	9406	9406	9406/S	9406/M	9406
		9406							
st_index	9446	9479/M	9479/S	9479	9479	9479	9479/S	9479/M	9479
		9479							
st_index	9584	9623/M	9623/S	9623	9623	9623	9623/S	9623/M	9623
		9623	9647/M	9647/S	9647	9647	9647	9647/S	9647/M
		9647	9647						
st_index	10157	10201/M	10201/S	10201	10201	10201	10201/S	10201/M	10201
		10201							
st_index	10408	10519/M	10519/S	10519	10519	10519	10519/S	10519/M	10519
		10519	10550/M	10550/S	10550	10550	10550	10550/S	10550/M
		10550	10550	10579/M	10579/S	10579	10579	10579	10579/S
		10579/M	10579	10579					
st_index	10879	10925/M	10925/S	10925	10925	10925	10925/S	10925/M	10925
		10925							
st_index	11095	11139/M	11139/S	11139	11139	11139	11139/S	11139/M	11139
		11139	11151/M	11151/S	11151	11151	11151	11151/S	11151/M
		11151	11151						
st_p	5633	5635/M							
st_p	6863	6941/M							
st_p	6871	6941/P	6993						
st_p	7354	7369/S	7370/S						
st_p	7377	7399							
st_p	7562	7585/M							
st_p	7568	7585/P	7587						
st_p	7730	7747/M							
st_p	7744	7747/P	7749/S	7756/S	7758/M	7759	7760/M		
st_p	7766	7820/M	7866/M	7898/M					
st_p	7786	7866/P	7866	7898/P	7898				
st_p	7796	7820/P	7821/P						
st_p	7932	7979/P	7979						
st_p	7932	7979/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cst_p	7988	8004/M							
cst_p	8002	8004/P	8006/M	8006					
cst_p	8011	8016/M							
cst_p	8014	8016/P	8018/S	8019/S	8022/S	8023/S	8024/M	8025	8026/M
cst_p	8044	8091/M							
cst_p	8064	8091/P	8092/M	8093	8094/M	8096			
cst_p	8110	8172/P	8172						
cst_p	8110	8172/M							
cst_p	8182	8227/P	8227						
cst_p	8182	8227/M							
cst_p	8381	8499/P	8499						
cst_p	8381	8499/M							
cst_p	8383	8425	8425	8493/P					
cst_p	8515	8533/S	8536/S	8543/M	8543	8544/P	8549/S	8550/S	8557
		8558/M	8559/M	8560	8561/M	8571/P	8575/P	8576/P	8576/P
cst_p	8581	8630/P	8630						
cst_p	8581	8630/M							
cst_p	8581	8635/S	8635/S						
cst_p	8583	8606	8608/P	8614/S	8617/P	8617/P	8619	8620/M	8623
		8624/S	8625/S	8627	8628	8630/P	8635/P	8642/M	8642
		8643/M	8643	8644/M	8647	8650	8675/P	8675/P	
cst_p	8682	8739/M	8739	8739	8739/M	8739	8739	8739	8739
		8739/P	8739/S	8739/S	8739	8739/S	8739/S	8739	8739/M
		8739	8739	8739/M	8739	8739	8739	8739	8739/P
		8739/P	8739/M	8739	8739	8739	8739/M	8739/M	8739/M
		8739/M	8739	8739	8739	8739	8739/P	8739/M	
cst_p	8682	8742							
cst_p	8682	8754	8754	8754/P					
cst_p	8684	8703	8708	8713/P	8723/P	8724	8728/M	8729/M	8729
		8730/M	8730	8731/M	8732/M	8733	8733	8738/M	8738
		8739/P	8740/M	8740	8741/P	8742/P	8742/P	8743/M	8744/M
		8744	8745/M	8746/M	8747	8748/M	8753/S	8754/P	
cst_p	8771	8803	8815/M	8816	8817/M	8825/M	8826	8827/M	
cst_p	8877	8896	8900	8901	8902	8903/M	8904	8905	8905
		8907/M	8908/M	8913	8916	8917/P	8917/P	8925/M	8926/M
		8933/M	8934	8935/M					
cst_p	8968	9000/M	9006	9007	9009/M	9018/S	9021/M	9021	9023/M
		9023	9024/M	9024	9033	9041	9049/M	9050	9051/M
cst_p	9059	9113	9121	9134/S	9139	9140	9140	9142	9144/P
		9145/M	9155/M	9155	9157/M	9157	9158/M	9158	9162
		9168	9174/M	9175	9176/M	9178/M	9188	9189/S	9190
		9192/S	9198/S	9199/M	9202	9203/M	9206/M		
cst_p	9349	9406/P	9406						
cst_p	9349	9406/M							
cst_p	9446	9479/P	9479						
cst_p	9446	9479/M							
cst_p	9584	9623/P	9623	9647/P	9647				
cst_p	9584	9623/M	9647/M						
cst_p	9662	9685/M	9685	9686	9687/M	9687	9688	9690	9690
		9694/P	9698/S	9699/S	9699	9701/S	9702/S	9702	9707/M
		9707	9707	9709/M	9709	9709	9712	9716	9717/P
		9717/P	9723/M	9723	9725	9726	9728/M	9730/M	9733/M
		9735/M	9742	9742	9744	9744	9745/P	9746/M	

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cst_p	9875	9882	9887	9894/P					
cst_p	9902	9921	9924	9924	9929	9930			
cst_p	9938	9956	9956	9961/M	9961	9962/M	9962	9963/M	9964/M
		10080/S	10081/P	10084/M	10128/M	10129/M	10129	10130/S	10131/M
		10131	10132/S	10133/S	10134/S				
cst_p	10157	10201/P	10201						
cst_p	10157	10201/M							
cst_p	10408	10466/M	10466	10466	10466/M	10466	10466	10466	10466
		10466/P	10466/S	10466/S	10466	10466/S	10466/S	10466	10466/M
		10466	10466	10466/M	10466	10466	10466	10466	10466/P
		10466/P	10466/M	10466	10466	10466	10466/M	10466/M	10466/M
		10466/M	10466	10466	10466	10466	10466/P	10466/M	
cst_p	10408	10519/P	10519	10550/P	10550	10579/P	10579		
cst_p	10408	10519/M	10550/M	10579/M					
cst_p	10408	10530	10530	10530/P					
cst_p	10410	10444/M	10459	10465	10466/P	10469	10470/M	10470	10471/M
		10472	10480	10481	10483/M	10488/M	10494/M	10495/S	10496/M
		10496/S	10497/S	10510	10511	10517/M	10521/M	10529/M	10530/P
		10539/M	10539	10540	10541/M	10559/M	10559	10565/M	10565
		10567	10568	10570/M	10571/P	10577	10578	10578	10579/P
		10583/P	10583/P	10587	10594	10594	10595	10595	10597/M
		10631	10632	10634/M	10635/M	10636/P	10636/P	10637/M	10637
		10638/M	10638	10639	10641	10642	10648	10648	10678/M
		10679/P	10679/P	10680/M	10680	10681/M	10681	10682	10685
		10686	10687	10693	10694	10694	10724/M	10726/M	10726
		10734	10735/M	10737/M	10739/M	10742/M	10743/M	10744/M	10745/M
		10746/M	10759/M	10765	10766	10767	10768/M	10768	10771/M
		10771	10775/M	10776/M	10776	10783/M	10784/M	10785/M	10786/M
		10787/M	10788/M	10789/S					
cst_p	10879	10925/P	10925						
cst_p	10879	10925/M							
cst_p	11095	11139/P	11139	11151/P	11151				
cst_p	11095	11139/M	11151/M						
cst_p	11096	11105	11110/S	11111/S	11113	11114/M	11114	11115/M	11116
		11118/M	11118	11122/M	11122	11128/M	11128	11130	11131
		11133/M	11134/P	11139/P	11145	11146/S	11148/S	11149/P	11151/P
		11152/M							
curr_ptlo	7394	7399/M	7402	7402	7403	7406	7407/M	7410/S	7411/S
curr_ptlo	8682	8742/M	8742	8742	8742	8742/M	8742	8742/S	8742/S
current_pit_value	9918	9924/M	9926	9927/M	9927	9929			
current_pit_value	9942	9956/M	9958	9959/M	9959	9961	9962		
dct_placement	6873	6894/M	6896/M	6899/M	6922	7019	7033	7040	7053
		7055	7057						
dct_placement	7766	7866/M	7866/M	7866/M	7866	7866	7866	7866	7866
		7866	7866	7898/M	7898/M	7898/M	7898	7898	7898
		7898	7898	7898	7898				
dct_placement	7932	7979/M	7979/M	7979/M	7979	7979	7979	7979	7979
		7979	7979						
dct_placement	8110	8172/M	8172/M	8172/M	8172	8172	8172	8172	8172
		8172	8172						
dct_placement	8182	8227/M	8227/M	8227/M	8227	8227	8227	8227	8227
		8227	8227						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dct_placement	8381	8499/M	8499/M	8499/M	8499	8499	8499	8499	8499
		8499	8499						
dct_placement	8581	8630/M	8630/M	8630/M	8630	8630	8630	8630	8630
		8630	8630						
dct_placement	9349	9406/M	9406/M	9406/M	9406	9406	9406	9406	9406
		9406	9406						
dct_placement	9446	9479/M	9479/M	9479/M	9479	9479	9479	9479	9479
		9479	9479						
dct_placement	9584	9623/M	9623/M	9623/M	9623	9623	9623	9623	9623
		9623	9623	9647/M	9647/M	9647	9647	9647	9647
		9647	9647	9647	9647				
dct_placement	10157	10201/M	10201/M	10201/M	10201	10201	10201	10201	10201
		10201	10201						
dct_placement	10408	10519/M	10519/M	10519/M	10519	10519	10519	10519	10519
		10519	10519	10550/M	10550/M	10550/M	10550	10550	10550
		10550	10550	10550	10550	10579/M	10579/M	10579/M	10579
		10579	10579	10579	10579	10579	10579		
dct_placement	10879	10925/M	10925/M	10925/M	10925	10925	10925	10925	10925
		10925	10925						
dct_placement	11095	11139/M	11139/M	11139/M	11139	11139	11139	11139	11139
		11139	11139	11151/M	11151/M	11151/M	11151	11151	11151
		11151	11151	11151	11151				
dcte	6676	6682/M	6684	6684	6685	6688/M	6689/M	6690/M	6691/M
		6695	6696/M	6697	6698/M	6700	6701/M	6704	6713
		6714/M	6716	6717/M	6719	6720/M	6724		
dcte	6746	6752/M	6754	6754	6755	6758/M	6759/M	6760	6761/M
		6763	6772	6773/M	6776				
dcte	6863	7054	7054	7054/M	7054	7054	7054/M	7054/M	7054/M
		7054	7054/M	7054	7054/M	7054	7054/M	7054/M	
dcte	6863	7056	7056	7056/M	7056	7056/M	7056/M	7056/M	7056
		7056/M	7056	7056/M	7056	7056/M	7056/M		
dcte	6863	7058	7058	7058/M	7058	7058/M	7058	7058/M	7058/M
dcte	6872	6969/M	6970	6972/M	6973/M	6974/M	6975/M	6981	6982/M
		6984	7004/M	7006	7008/M	7009/M	7010/M	7011/M	7015
		7020/S	7021/S	7026	7026	7027/M	7029/M	7031/M	7034/S
		7035/S	7036/M	7038/M	7041/S	7043/M	7054/P	7056/P	7058/P
		7062							
dcte	7072	7093	7109	7110/M	7130	7130	7131/M	7134/M	7136/M
		7159	7160/M	7162	7163/M	7165	7166/M	7172/M	
dcte	7184	7205	7221	7222/M	7226	7227/M	7242/M	7244/M	7268
		7269/M	7271	7272/M	7274	7275/M	7281/M		
dcte	7293	7312	7323	7324/M	7326	7327/M	7329	7330/M	7347/M
dcte	7766	7864/M	7864	7864	7864	7864/M	7864/M	7864/M	7864/M
		7864	7864/M	7864	7864/M	7864	7864/M	7864	7864
		7864/M	7864	7864/M	7864	7864/M	7864		
dcte	7766	7866/M	7866	7866/M	7866/M	7866/M	7866/M	7866	7866/M
		7866	7866/M	7866	7866/M	7866/M	7866/M	7866/M	7866
		7866/S	7866/S	7866	7866	7866/M	7866/M	7866/M	7866/S
		7866/S	7866/M	7866/M	7866/S	7866/M	7866/P	7866/P	7866/P
		7866	7898/M	7898	7898/M	7898/M	7898/M	7898/M	7898
		7898/M	7898	7898/M	7898	7898/M	7898/M	7898/M	7898/M
		7898	7898/S	7898/S	7898	7898	7898/M	7898/M	7898/M
		7898/S	7898/S	7898/M	7898/M	7898/S	7898/M	7898/P	7898/P

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7898/P	7898						
dcte	7766	7866	7866	7866/M	7866	7866	7866/M	7866/M	7866/M
		7866	7866/M	7866	7866/M	7866	7866/M	7866/M	7898
		7898	7898/M	7898	7898	7898/M	7898/M	7898/M	7898
		7898/M	7898	7898/M	7898	7898/M	7898/M		
dcte	7766	7866	7866	7866/M	7866	7866/M	7866/M	7866/M	7866
		7866/M	7866	7866/M	7866	7866/M	7866/M	7898	7898
		7898/M	7898	7898/M	7898/M	7898/M	7898	7898/M	7898
		7898/M	7898	7898/M	7898/M				
dcte	7766	7866	7866	7866/M	7866	7866/M	7866	7866/M	7866/M
		7898	7898	7898/M	7898	7898/M	7898	7898/M	7898/M
dcte	7932	7979/M	7979	7979/M	7979/M	7979/M	7979/M	7979	7979/M
		7979	7979/M	7979	7979/M	7979/M	7979/M	7979/M	7979
		7979/S	7979/S	7979	7979	7979/M	7979/M	7979/M	7979/S
		7979/S	7979/M	7979/M	7979/S	7979/M	7979/P	7979/P	7979/P
		7979							
dcte	7932	7979	7979	7979/M	7979	7979	7979/M	7979/M	7979/M
		7979	7979/M	7979	7979/M	7979	7979/M	7979/M	
dcte	7932	7979	7979	7979/M	7979	7979/M	7979/M	7979/M	7979
		7979/M	7979	7979/M	7979	7979/M	7979/M		
dcte	7932	7979	7979	7979/M	7979	7979/M	7979	7979/M	7979/M
dcte	8110	8172/M	8172	8172/M	8172/M	8172/M	8172/M	8172	8172/M
		8172	8172/M	8172	8172/M	8172/M	8172/M	8172/M	8172
		8172/S	8172/S	8172	8172	8172/M	8172/M	8172/M	8172/S
		8172/S	8172/M	8172/M	8172/S	8172/M	8172/P	8172/P	8172/P
		8172							
dcte	8110	8172	8172	8172/M	8172	8172	8172/M	8172/M	8172/M
		8172	8172/M	8172	8172/M	8172	8172/M	8172/M	
dcte	8110	8172	8172	8172/M	8172	8172/M	8172/M	8172/M	8172
		8172/M	8172	8172/M	8172	8172/M	8172/M		
dcte	8110	8172	8172	8172/M	8172	8172/M	8172	8172/M	8172/M
dcte	8182	8227/M	8227	8227/M	8227/M	8227/M	8227/M	8227	8227/M
		8227	8227/M	8227	8227/M	8227/M	8227/M	8227/M	8227
		8227/S	8227/S	8227	8227	8227/M	8227/M	8227/M	8227/S
		8227/S	8227/M	8227/M	8227/S	8227/M	8227/P	8227/P	8227/P
		8227							
dcte	8182	8227	8227	8227/M	8227	8227	8227/M	8227/M	8227/M
		8227	8227/M	8227	8227/M	8227	8227/M	8227/M	
dcte	8182	8227	8227	8227/M	8227	8227/M	8227/M	8227/M	8227
		8227/M	8227	8227/M	8227	8227/M	8227/M		
dcte	8182	8227	8227	8227/M	8227	8227/M	8227	8227/M	8227/M
dcte	8381	8499/M	8499	8499/M	8499/M	8499/M	8499/M	8499	8499/M
		8499	8499/M	8499	8499/M	8499/M	8499/M	8499/M	8499
		8499/S	8499/S	8499	8499	8499/M	8499/M	8499/M	8499/S
		8499/S	8499/M	8499/M	8499/S	8499/M	8499/P	8499/P	8499/P
		8499							
dcte	8381	8499	8499	8499/M	8499	8499	8499/M	8499/M	8499/M
		8499	8499/M	8499	8499/M	8499	8499/M	8499/M	
dcte	8381	8499	8499	8499/M	8499	8499/M	8499/M	8499/M	8499
		8499/M	8499	8499/M	8499	8499/M	8499/M		
dcte	8381	8499	8499	8499/M	8499	8499/M	8499	8499/M	8499/M
dcte	8581	8630/M	8630	8630/M	8630/M	8630/M	8630/M	8630	8630/M
		8630	8630/M	8630	8630/M	8630/M	8630/M	8630/M	8630

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8630/S	8630/S	8630	8630	8630/M	8630/M	8630/M	8630/S
		8630/S	8630/M	8630/M	8630/S	8630/M	8630/P	8630/P	8630/P
		8630							
dcte	8581	8630	8630	8630/M	8630	8630	8630/M	8630/M	8630/M
		8630	8630/M	8630	8630/M	8630	8630/M	8630/M	
dcte	8581	8630	8630	8630/M	8630	8630/M	8630/M	8630/M	8630
		8630/M	8630	8630/M	8630	8630/M	8630/M		
dcte	8581	8630	8630	8630/M	8630	8630/M	8630	8630/M	8630/M
dcte	9349	9406/M	9406	9406/M	9406/M	9406/M	9406/M	9406	9406/M
		9406	9406/M	9406	9406/M	9406/M	9406/M	9406/M	9406
		9406/S	9406/S	9406	9406	9406/M	9406/M	9406/M	9406/S
		9406/S	9406/M	9406/M	9406/S	9406/M	9406/P	9406/P	9406/P
		9406							
dcte	9349	9406	9406	9406/M	9406	9406	9406/M	9406/M	9406/M
		9406	9406/M	9406	9406/M	9406	9406/M	9406/M	
dcte	9349	9406	9406	9406/M	9406	9406/M	9406/M	9406/M	9406
		9406/M	9406	9406/M	9406	9406/M	9406/M		
dcte	9349	9406	9406	9406/M	9406	9406/M	9406	9406/M	9406/M
dcte	9349	9417/M	9417	9417	9417	9417/M	9417/M	9417/M	9417/M
		9417	9417/M	9417	9417/M	9417	9417/M	9417	9417
		9417/M	9417	9417/M	9417	9417/M	9417		
dcte	9446	9479/M	9479	9479/M	9479/M	9479/M	9479/M	9479	9479/M
		9479	9479/M	9479	9479/M	9479/M	9479/M	9479/M	9479
		9479/S	9479/S	9479	9479	9479/M	9479/M	9479/M	9479/S
		9479/S	9479/M	9479/M	9479/S	9479/M	9479/P	9479/P	9479/P
		9479							
dcte	9446	9479	9479	9479/M	9479	9479	9479/M	9479/M	9479/M
		9479	9479/M	9479	9479/M	9479	9479/M	9479/M	
dcte	9446	9479	9479	9479/M	9479	9479/M	9479/M	9479/M	9479
		9479/M	9479	9479/M	9479	9479/M	9479/M		
dcte	9446	9479	9479	9479/M	9479	9479/M	9479	9479/M	9479/M
dcte	9491	9528/M	9528	9528	9528	9528/M	9528/M	9528/M	9528/M
		9528	9528/M	9528	9528/M	9528	9528/M	9528	9528
		9528/M	9528	9528/M	9528	9528/M	9528	9562/M	9562
		9562	9562	9562/M	9562/M	9562/M	9562	9562	9562/M
		9562	9562/M	9562	9562/M	9562	9562	9562/M	9562
		9562/M	9562	9562/M	9562				
dcte	9584	9623/M	9623	9623/M	9623/M	9623/M	9623/M	9623	9623/M
		9623	9623/M	9623	9623/M	9623/M	9623/M	9623/M	9623
		9623/S	9623/S	9623	9623	9623/M	9623/M	9623/M	9623/S
		9623/S	9623/M	9623/M	9623/S	9623/M	9623/P	9623/P	9623/P
		9623	9647/M	9647	9647/M	9647/M	9647/M	9647/M	9647
		9647/M	9647	9647/M	9647	9647/M	9647/M	9647/M	9647/M
		9647	9647/S	9647/S	9647	9647	9647/M	9647/M	9647/M
		9647/S	9647/S	9647/M	9647/M	9647/S	9647/M	9647/P	9647/P
		9647/P	9647						
dcte	9584	9623	9623	9623/M	9623	9623	9623/M	9623/M	9623/M
		9623	9623/M	9623	9623/M	9623	9623/M	9623/M	9647
		9647	9647/M	9647	9647	9647/M	9647/M	9647/M	9647
		9647/M	9647	9647/M	9647	9647/M	9647/M		
dcte	9584	9623	9623	9623/M	9623	9623/M	9623/M	9623/M	9623
		9623/M	9623	9623/M	9623	9623/M	9623/M	9647	9647
		9647/M	9647	9647/M	9647/M	9647/M	9647	9647/M	9647

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9647/M	9647	9647/M	9647/M				
dcte	9584	9623	9623	9623/M	9623	9623/M	9623	9623/M	9623/M
		9647	9647	9647/M	9647	9647/M	9647	9647/M	9647/M
dcte	10157	10198/M	10198	10198	10198	10198/M	10198/M	10198/M	10198/M
		10198	10198/M	10198	10198/M	10198	10198/M	10198	10198
		10198/M	10198	10198/M	10198	10198/M	10198		
dcte	10157	10201/M	10201	10201/M	10201/M	10201/M	10201/M	10201	10201/M
		10201	10201/M	10201	10201/M	10201/M	10201/M	10201/M	10201
		10201/S	10201/S	10201	10201	10201/M	10201/M	10201/M	10201/S
		10201/S	10201/M	10201/M	10201/S	10201/M	10201/P	10201/P	10201/P
		10201							
dcte	10157	10201	10201	10201/M	10201	10201	10201/M	10201/M	10201/M
		10201	10201/M	10201	10201/M	10201	10201/M	10201/M	
dcte	10157	10201	10201	10201/M	10201	10201/M	10201/M	10201/M	10201
		10201/M	10201	10201/M	10201	10201/M	10201/M		
dcte	10157	10201	10201	10201/M	10201	10201/M	10201	10201/M	10201/M
dcte	10408	10519/M	10519	10519/M	10519/M	10519/M	10519/M	10519	10519/M
		10519	10519/M	10519	10519/M	10519/M	10519/M	10518/M	10519
		10519/S	10519/S	10519	10519	10519/M	10519/M	10519/M	10519/S
		10519/S	10519/M	10519/M	10519/S	10519/M	10519/P	10519/P	10519/P
		10519	10550/M	10550	10550/M	10550/M	10550/M	10550/M	10550
		10550/M	10550	10550/M	10550	10550/M	10550/M	10550/M	10550/M
		10550	10550/S	10550/S	10550	10550	10550/M	10550/M	10550/M
		10550/S	10550/S	10550/M	10550/M	10550/S	10550/M	10550/P	10550/P
		10550/P	10550	10579/M	10579	10579/M	10579/M	10579/M	10579/M
		10579	10579/M	10579	10579/M	10579	10579/M	10579/M	10579/M
		10579/M	10579	10579/S	10579/S	10579	10579	10579/M	10579/M
		10579/M	10579/S	10579/S	10579/M	10579/M	10579/S	10579/M	10579/P
		10579/P	10579/P	10579					
dcte	10408	10519	10519	10519/M	10519	10519	10519/M	10519/M	10519/M
		10519	10519/M	10519	10519/M	10519	10519/M	10519/M	10550
		10550	10550/M	10550	10550	10550/M	10550/M	10550/M	10550
		10550/M	10550	10550/M	10550	10550/M	10550/M	10579	10579
		10579/M	10579	10579	10579/M	10579/M	10579/M	10579	10579/M
		10579	10579/M	10579	10579/M	10579/M			
dcte	10408	10519	10519	10519/M	10519	10519/M	10519/M	10519/M	10519
		10519/M	10519	10519/M	10519	10519/M	10519/M	10550	10550
		10550/M	10550	10550/M	10550/M	10550/M	10550	10550/M	10550
		10550/M	10550	10550/M	10550/M	10579	10579	10579/M	10579
		10579/M	10579/M	10579/M	10579	10579/M	10579	10579/M	10579
		10579/M	10579/M						
dcte	10408	10519	10519	10519/M	10519	10519/M	10519	10519/M	10519/M
		10550	10550	10550/M	10550	10550/M	10550	10550/M	10550/M
		10579	10579	10579/M	10579	10579/M	10579	10579/M	10579/M
dcte	10408	10725/M	10725	10725	10725	10725/M	10725/M	10725/M	10725/M
		10725	10725/M	10725	10725/M	10725	10725/M	10725	10725
		10725/M	10725	10725/M	10725	10725/M	10725		
dcte	10879	10925/M	10925	10925/M	10925/M	10925/M	10925/M	10925	10925/M
		10925	10925/M	10925	10925/M	10925/M	10925/M	10925/M	10925
		10925/S	10925/S	10925	10925	10925/M	10925/M	10925/M	10925/S
		10925/S	10925/M	10925/M	10925/S	10925/M	10925/P	10925/P	10925/P
		10925							
dcte	10879	10925	10925	10925/M	10925	10925	10925/M	10925/M	10925/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10925	10925/M	10925	10925/M	10925	10925/M	10925/M	
dcte	10879	10925	10925	10925/M	10925	10925/M	10925/M	10925/M	10925
		10925/M	10925	10925/M	10925	10925/M	10925/M		
dcte	10879	10925	10925	10925/M	10925	10925/M	10925	10925/M	10925/M
dcte	10964	10997/M	10997	10997	10997	10997/M	10997/M	10997	10997/M
		10997	10997	10997/M	10997				
dcte	11095	11139/M	11139	11139/M	11139/M	11139/M	11139/M	11139	11139/M
		11139	11139/M	11139	11139/M	11139/M	11139/M	11139/M	11139
		11139/S	11139/S	11139	11139	11139/M	11139/M	11139/M	11139/S
		11139/S	11139/M	11139/M	11139/S	11139/M	11139/P	11139/P	11139/P
		11139	11151/M	11151	11151/M	11151/M	11151/M	11151/M	11151
		11151/M	11151	11151/M	11151	11151/M	11151/M	11151/M	11151/M
		11151	11151/S	11151/S	11151	11151	11151/M	11151/M	11151/M
		11151/S	11151/S	11151/M	11151/M	11151/S	11151/M	11151/P	11151/P
		11151/P	11151						
dcte	11095	11139	11139	11139/M	11139	11139	11139/M	11139/M	11139/M
		11139	11139/M	11139	11139/M	11139	11139/M	11139/M	11151
		11151	11151/M	11151	11151	11151/M	11151/M	11151/M	11151
		11151/M	11151	11151/M	11151	11151/M	11151/M		
dcte	11095	11139	11139	11139/M	11139	11139/M	11139/M	11139/M	11139
		11139/M	11139	11139/M	11139	11139/M	11139/M	11151	11151
		11151/M	11151	11151/M	11151/M	11151/M	11151	11151/M	11151
		11151/M	11151	11151/M	11151/M				
dcte	11095	11139	11139	11139/M	11139	11139/M	11139	11139/M	11139/M
		11151	11151	11151/M	11151	11151/M	11151	11151/M	11151/M
debug_index	3146	10844/M							
debug_list_pointer	3149	10840	10840	10841/M					
debug_mask_register	3148	10842/M							
delayed_swapin_work	1297	9460	9598	10173/M	10173	10236/M	10236		
dest	5600	5610							
dest	9936	10106							
determine_dispatching_priority	10310	10019	10046	10070	10346	10392			
dfc$command_record_bytes	1367	1375							
dfc$division_overwrite_words	1354	1382							
dfc$esm_command_record_size	1375	1383							
dfc$esm_header_record_size	1376	1383							
dfc$esm_maintenance_buf_size	1355	1386							
dfc$esm_memory_base_shift	1361	1383	1384	1384					
dfc$header_record_bytes	1366	1376							
dfc$max_esm_memory_size	1356	1385							
dfc$max_number_of_mainframes	1363	1348							
dfc$min_data_record_bytes	1371	1382							
dfc$min_esm_division_size	1381	1385							
dft$mainframe_set	1348	1298	1299	1437	1438				
dispatch_control	1159	6990/M	7758/M	7866/M	7898/M	7979/M	8024/M	8092/M	8172/M
		8227/M	8499/M	8559/M	8630/M	8746/M	8815/M	8825/M	8933/M
		9049/M	9174/M	9406/M	9479/M	9623/M	9647/M	10084/M	10201/M
		10519/M	10550/M	10579/M	10925/M	11139/M	11151/M		
dispatching_control	1280	6885/S	7819	7822	7866/S	7898/S	7979/S	8172/S	8227/S
		8497/S	8499/S	8625/S	8630/S	8739	8739	8739/M	8739
		8838	8916	9406/S	9479/S	9623/S	9647/S	9712	9716
		9723/M	9723	9994/M	10000/M	10002/M	10004/M	10008	10010
		10017/M	10027/M	10029/M	10030/M	10031/M	10032/M	10043/M	10044/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10066/M	10068/M	10179	10182	10194/S	10201/S	10275	10321
		10323	10324/M	10324	10327	10328	10330/S	10332/M	10332
		10335/M	10337/S	10343	10379	10389/M	10391/M	10466	10466
		10466/M	10466	10497/S	10519/S	10550/S	10579/S	10925/S	11111/S
		11139/S	11151/S						
dispatching_control	2582	6885	7866	7898	7979	8172	8227	8497	8499
		8625	8630	9406	9479	9623	9647	10045	10057
		10194	10201	10278	10330	10337	10374	10497	10519
		10550	10579	10925	11111	11139	11151		
dispatching_control_index	1449	6885/S	7866/S	7898/S	7979/S	8172/S	8227/S	8497/S	8499/S
		8625/S	8630/S	9406/S	9479/S	9623/S	9647/S	10027/M	10043/M
		10066/M	10194/S	10201/S	10330/S	10337/S	10379	10391/M	10498/S
		10519/S	10550/S	10579/S	10925/S	11111/S	11139/S	11151/S	
dispatching_control_index	4903	10057/S	10067						
dispatching_control_index	10269	10280	10282/S	10283/S	10285/S				
dispatching_control_info	4891	7818/M	9991	10000	10002	10017	10057/S	10059	10060
		10066	10068						
dispatching_control_p	10268	10277/M	10282	10283	10285				
dispatching_control_p	10369	10373/M	10386	10388	10389	10396			
dispatching_controls	9943	10056/M	10058	10058	10060				
dispatching_priority	1145	8739/P	8739/S	8739/S	8739/S	8739/S	9694/P	9698/S	9699/S
		9701/S	9702/S	10466/P	10466/S	10466/S	10466/S	10466/S	10483/M
		10488/M	10495/S	10742/M	10787/M	10789/S			
dispatching_priority	1450	7819	7822	8838	8916	10002/M	10008	10010	10029/M
		10179	10182	10321	10324/M	10332/M	10335/M	10343	
dispatching_priority	1464	10060	10331	10338					
dispatching_priority	2530	6626	6912	6915	7866	7866	7898	7898	7979
		7979	8172	8172	8227	8227	8445	8499	8499
		8608	8623/M	8623	8630	8630	8813	8818/S	8828/S
		8834	8913	9188	9190	9406	9406	9479	9479
		9623	9623	9647	9647	10190/M	10199	10201	10201
		10203	10519	10519	10550	10550	10579	10579	10925
		10925	11139	11139	11151	11151			
dispatching_priority	4904	9991	10001	10003	10018	10061			
dispatching_priority	4942	6626/M	6649/M	6682/S	6693	6724/S	6913/M	6915/M	6919
		6920/M	6929	6930/M	6940/S	6969/S	6978	6984/S	6989/S
		7004/S	7013	7015/S	7062/S	7864/S	7864	7864/S	7866/M
		7866/M	7866	7866/M	7866	7866/M	7866/S	7866/S	7866
		7866/S	7866/S	7866/S	7866	7866/S	7866/S	7898/M	7898/M
		7898	7898/M	7898	7898/M	7898/S	7898/S	7898	7898/S
		7898/S	7898/S	7898	7898/S	7898/S	7979/M	7979/M	7979
		7979/M	7979	7979/M	7979/S	7979/S	7979	7979/S	7979/S
		7979/S	7979	7979/S	7979/S	8172/M	8172/M	8172	8172/M
		8172	8172/M	8172/S	8172/S	8172	8172/S	8172/S	8172/S
		8172	8172/S	8172/S	8227/M	8227/M	8227	8227/M	8227
		8227/M	8227/S	8227/S	8227	8227/S	8227/S	8227/S	8227
		8227/S	8227/S	8445/M	8499/M	8499/M	8499	8499/M	8499
		8499/M	8499/S	8499/S	8499	8499/S	8499/S	8499/S	8499
		8499/S	8499/S	8544/M	8608/M	8630/M	8630/M	8630	8630/M
		8630	8630/M	8630/S	8630/S	8630	8630/S	8630/S	8630/S
		8630	8630/S	8630/S	8741/M	8813/M	8834/M	8913/M	9406/M
		9406/M	9406	9406/M	9406	9406/M	9406/S	9406/S	9406
		9406/S	9406/S	9406/S	9406	9406/S	9406/S	9417/S	9417

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9417/S	9479/M	9479/M	9479	9479/M	9479	9479/M	9479/S
		9479/S	9479	9479/S	9479/S	9479/S	9479	9479/S	9479/S
		9528/S	9528	9528/S	9562/S	9562	9562/S	9623/M	9623/M
		9623	9623/M	9623	9623/M	9623/S	9623/S	9623	9623/S
		9623/S	9623/S	9623	9623/S	9623/S	9647/M	9647/M	9647
		9647/M	9647	9647/M	9647/S	9647/S	9647	9647/S	9647/S
		9647/S	9647	9647/S	9647/S	10198/S	10198	10198/S	10199/M
		10201/M	10201/M	10201	10201/M	10201	10201/M	10201/S	10201/S
		10201	10201/S	10201/S	10201/S	10201	10201/S	10201/S	10203/M
		10519/M	10519/M	10519	10519/M	10519	10519/M	10519/S	10519/S
		10519	10519/S	10519/S	10519/S	10519	10519/S	10519/S	10550/M
		10550/M	10550	10550/M	10550	10550/M	10550/S	10550/S	10550
		10550/S	10550/S	10550/S	10550	10550/S	10550/S	10579/M	10579/M
		10579	10579/M	10579	10579/M	10579/S	10579/S	10579	10579/S
		10579/S	10579/S	10579	10579/S	10579/S	10725/S	10725	10725/S
		10787	10925/M	10925/M	10925	10925/M	10925	10925/M	10925/S
		10925/S	10925	10925/S	10925/S	10925/S	10925	10925/S	10925/S
		10984	11003/M	11086/M	11139/M	11139/M	11139	11139/M	11139
		11139/M	11139/S	11139/S	11139	11139/S	11139/S	11139/S	11139
		11139/S	11139/S	11151/M	11151/M	11151	11151/M	11151	11151/M
		11151/S	11151/S	11151	11151/S	11151/S	11151/S	11151	11151/S
		11151/S							
dispatching_priority	8682	8739	8739	8739	8739/S	8739	8739/P	8739/S	8739/S
		8739	8739	8739/S	8739	8739	8739	8739	8739/P
		8739/S	8739/S						
dispatching_priority	9661	9694	9694	9694	9694/S	9694	9694/P	9694/S	9694/S
		9694	9694	9694/S	9694	9694	9694	9694	9694/P
		9694/S	9694/S						
dispatching_priority	9755	9760	9773	9774	9777/S	9780	9782/P	9784/S	9785/S
		9790	9791	9794/S	9797	9800	9801	9804	9807/P
		9809/S	9810/S						
dispatching_priority	10167	10179/M	10182/M	10185	10186/M	10187	10188/M	10190	
dispatching_priority	10408	10466	10466	10466	10466/S	10466	10466/P	10466/S	10466/S
		10466	10466	10466/S	10466	10466	10466	10466	10466/P
		10466/S	10466/S	10774	10774	10774	10774/S	10774	10774/P
		10774/S	10774/S	10774	10774	10774/S	10774	10774	10774
		10774	10774/P	10774/S	10774/S				
dispatching_priority	10955	11003							
dispatching_priority_bias	2532	10180	10183						
dispatching_priority_bias_id	2531	10177	10178						
dispatching_priority_integer	1184	6947	6953	6988/M	7760/M	7866	7866	7866/M	7898
		7898	7898/M	7979	7979	7979/M	8026/M	8094/M	8172
		8172	8172/M	8227	8227	8227/M	8499	8499	8499/M
		8561/M	8630	8630	8630/M	8748/M	8817/M	8827/M	8935/M
		9051/M	9176/M	9406	9406	9406/M	9479	9479	9479/M
		9623	9623	9623/M	9647	9647	9647/M	10201	10201
		10201/M	10519	10519	10519/M	10521/M	10541/M	10550	10550
		10550/M	10579	10579	10579/M	10724/M	10743/M	10925	10925
		10925/M	11139	11139	11139/M	11151	11151	11151/M	
dispatching_priority_time	4797	8739	8739/M	8739	8739	8739/M	8739	9694	9694/M
		9694	9694	9694/M	9694	9776	9784/M	9785	9793
		9809/M	9810	10466	10466/M	10466	10466	10466/M	10466
		10774	10774/M	10774	10774	10774/M	10774		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dispatching_timeslice	1466	6885	7866	7898	7979	8172	8227	8497	8499
		8626	8630	9406	9479	9623	9647	10195	10201
		10498	10519	10550	10579	10925	11112	11139	11151
dmt$system_file_id	1397	1330							
dp	6405	10673	10673	10674/S	10711	10713	10715		
dp	8682	8739	8739/P						
dp	8682	8739	8739/S	8739	8739/S	8739	8739/S		
dp	9661	9694	9694/P						
dp	9661	9694	9694/S	9694	9694/S	9694	9694/S		
dp	9753	9768	9768/S	9782	9782/S	9807	9807/S		
dp	9758	9767	9768/P						
dp	9827	9834	9841/S						
dp	10408	10466	10466/P	10774	10774/P				
dp	10408	10466	10466/S	10466	10466/S	10466	10466/S	10774	10774/S
		10774	10774/S	10774	10774/S				
dp_mask	6414	10673	10673	10674/S	10711	10713	10715		
dp_trick_conversion	6409	10428							
dpc$console_row_size	5883	5877							
dpc$top_line_message_size	5877	5872							
dpt$top_line_message	5872	5847							
dual_state_jps	1154	6994	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10519	10550	10579
		10595	10925	11139	11151				
dual_state_prior_subpriority	1146	10494/M	10597/M	10744/M	10788/M				
dual_state_priority	3791	10744/M							
end_of_wait_time	4945	6632/M	8023/M	8286	8288	8288	8288	8321	8327
		8327	8445/M	8608/M	8930/M	9047/M	9173/M	9261	9262
		10550	10550	10561	10561	10561	10750	10816	10896
		10896	11010/M	11124	11124	11124			
end_of_wait_time	8276	8288/M	8288	8288					
end_of_wait_time	8317	8321/M	8327	8327					
end_of_wait_time	10408	10561/M	10561	10561					
end_of_wait_time	11095	11124/M	11124	11124					
end_ptlo	9356	9396/M	9398	9398					
enforce_maximums	4789	8739	9694	9801	10466	10774			
enforce_maximums	4815	6694	6694/M	6979	6979/M	7014	7014/M	7864	7864/M
		7866	7866/M	7866	7866/M	7898	7898/M	7898	7898/M
		7979	7979/M	7979	7979/M	8172	8172/M	8172	8172/M
		8227	8227/M	8227	8227/M	8499	8499/M	8499	8499/M
		8630	8630/M	8630	8630/M	8739/M	8739	8739/M	8739
		8739/M	8739	8739/M	8739	8739/M	8739/M	8739	8739
		8739/M	8739	8739/M	9406	9406/M	9406	9406/M	9417
		9417/M	9479	9479/M	9479	9479/M	9528	9528/M	9562
		9562/M	9623	9623/M	9623	9623/M	9647	9647/M	9647
		9647/M	9694/M	9694	9694/M	9694	9694/M	9694	9694/M
		9694	9694/M	9694/M	9694	9694	9694/M	9694	9694/M
		9765/M	9766	9766/M	9768	9768/M	9781	9781/M	9782
		9782/M	9802/M	9803	9806	9806/M	9807	9807/M	9835
		9836/M	9858	9860/M	10198	10198/M	10201	10201/M	10201
		10201/M	10466/M	10466	10466/M	10466	10466/M	10466	10466/M
		10466	10466/M	10466/M	10466	10466	10466/M	10466	10466/M
		10519	10519/M	10519	10519/M	10550	10550/M	10550	10550/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10579	10579/M	10579	10579/M	10658	10660/M	10725	10725/M
		10774/M	10774	10774/M	10774	10774/M	10774	10774/M	10774
		10774/M	10774/M	10774	10774	10774/M	10774	10774/M	10925
		10925/M	10925	10925/M	11139	11139/M	11139	11139/M	11151
		11151/M	11151	11151/M					
entry_status	1266	5364	5369/M	5694	7459	7505	7539	7577	7668
		7695	7698	7823	8471	8471/M	9197	9332	9372
		10081	10081/M	10568	11131				
environment	5986	8478	8479	8652	8653				
excess_cp_time_used	8682	8739/M	8739/P						
excess_cp_time_used	9675	9716/M	9717/P						
excess_cp_time_used	10408	10466/M	10466/P						
excess_service_used	10365	10376							
executing_task_count	1283	6967/M	6967	6992/M	6992	7865	7866/M	7866	7866/M
		7866	7897	7898/M	7898	7898/M	7898	7977	7979/M
		7979	7979/M	7979	8171	8172/M	8172	8172/M	8172
		8227/M	8227	8227/M	8227	8498	8499/M	8499	8499/M
		8499	8543/M	8543	8628	8630/M	8630	8630/M	8630
		8740/M	8740	9405	9406/M	9406	9406/M	9406	9478
		9479/M	9479	9479/M	9479	9622	9623/M	9623	9623/M
		9623	9646	9647/M	9647	9647/M	9647	10200	10201/M
		10201	10201/M	10201	10513/M	10513	10519/M	10519	10519/M
		10519	10539/M	10539	10550	10550/M	10550	10550/M	10550
		10579/M	10579	10579/M	10579	10693	10726/M	10726	10924
		10925/M	10925	10925/M	10925	11118/M	11118	11139/M	11139
		11139/M	11139	11147/M	11147	11151/M	11151	11151/M	11151
expected_wait_time	4845	9033	9035	9041					
expected_wait_time	4858	9113	9115	9121					
expired_dispatching_control	10366	10378							
f1	8407	8474							
f2	8408	8475							
fde_p	5236	5258/M	5259						
fde_p	5286	5293/M	5293						
fde_p	5288	5293/P	5295/P						
fde_p	8182	8222/P	8222/P	8232/P	8232/P				
fde_p	8182	8222/M	8222	8232/M	8232				
fde_p	8207	8222/P	8223/P	8232/P	8233	8249/P			
file_entry_index	1402	5248	5293	8222	8232				
file_hash	1404	5247	5259	5293	5293	8222	8222	8232	8232
file_hash	3884	5259	5293	8222	8232				
find_dispatching_control_set	10384	10384	10394	10402					
find_next_job	7690	7690	7704	7710					
find_next_task	10672	10672	10692	10698	10702	10722			
flags	3105	8484/M	8484	8486/M	8486	8669/M	8669	8671/M	8671
forward_link	2032	11034/M							
found	8208	8216/M	8234	8234	8236/M	8245			
free_ptl	6645	6657	8544	8741	11086				
free_running_clock	10408	10550/M	10550	10550					
free_running_clock	10887	10893/M	10896	10896					
fwa	44	53							
gfc$fde_size	5275	5248	5293	8222	8232				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gfc$fde_table_base	5273	5248	5274	5293	8222	8232			
gfc$fk_catalog	3960	3972							
gfc$fk_job_local_file	3962	3971							
gfc$fm_mass_storage_file	3975	3900							
gfc$fm_served_file	3976	3903							
gfc$monitor_interlocks	5307	5294	8222	8232					
gfc$tr_job	1415	5251	5293	8222	8232				
gfc$tr_system	1415	5250	5293	8222	8232				
gfp$mtr_get_fde_p	5234	5263	5293	8222	8232				
gfp$mtr_get_locked_fde_p	5286	5298	8222	8232					
gft$allocation_unit_size	3934	3889							
gft$attach_count	3939	3880	3881						
gft$fde_flags	3909	3877							
gft$file_desc_entry_p	5280	5236							
gft$file_descriptor_entry	3874	3866	3879	5280					
gft$file_descriptor_index	1412	1402							
gft$file_kind	3956	3883	3968						
gft$file_media	3975	3899							
gft$locked_file_desc_entry_p	3866	5288	8207						
gft$open_count	3993	3882	4009						
gft$queue_status	4004	3892							
gft$segment_lock_info	4008	3885							
gft$signature_lock	3981	3875							
gft$system_file_identifier	1401	1397	2101	2558	2735	2943	5234	5286	5974
gft$table_residence	1415	1403	5244						
gft$transfer_unit_size	3945	3890							
global_task_id	2519	7056	7260	7866	7898	7979	8172	8227	8454/M
		8499	8630	8646/M	8837/S	8839/S	8842/S	8844/S	8855/P
		8919/S	8921/S	9006	9189/S	9192/S	9406	9479	9623
		9647	10181	10201	10519	10550	10579	10925	11139
		11151							
global_task_id	6559	6562/S							
global_taskid	4859	9101	9102/P	9187	9194/P	9196/S			
gtid	10956	10984/S	10996	11006					
gtid	10971	11018/M	11019/M	11020/P					
guaranteed_service_quantum	2575	9200							
hash	5241	5247/M	5254/M						
hash	5286	5293/M	5293/M						
hash	8182	8222/M	8222/M	8232/M	8232/M				
head	2170	6630/M	7877	7877/S	7877/M	7877	7877/M	7877	7877/M
		8074	8083/M	8086/M	8135	8142	8143/M	8146/M	8146
		8147/S	8148/M	8223	8223/S	8223/M	8223	8223/M	8223
		8223/M	8234	8234	8235	8239/S	8249	8249/S	8249/M
		8249	8249/M	8249	8249/M	8268	8269/S	8288	8288
		8288/M	8288	8288/M	8288	8288/M	8288/M	8319	8323
		8324/M	8330	8337/M	8338	8339/M	8341/M	8362	8365/S
		8369/M	8369	8371/M	8371	8374/M	8445/M	8608/M	10550/S
		10550	10550	10550/M	10550	10550/S	10550/M	10550/S	10550
		10550	10561	10561	10561/M	10561	10561/M	10561	10561/M
		10561/M	10747	10750/S	10813	10816/S	10896/S	10896/S	10896
		10897	10898	10900/M	10900	10901/S	10902/M	10932	11008/M
		11124	11124	11124/M	11124	11124/M	11124	11124/M	11124/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
i	9944	10107	10108/S	10108					
i	10970	10983	10984/S	10995	10996/S	11002/S	11003/S	11006/S	
i#disable_traps	5579	5596							
i#move	5599	5615	10106						
i#program_error	5283	5260	5293	6448	6494	6502	7454	7466	7512
		7547	7599	7725	7757	7927	7983	8039	8104
		8176	8222	8232	8270	8292	8451	8508	8553
		8610	8677	8756	8870	8961	9010	9028	9048
		9104	9211	9250	9257	9271	9339	9486	9539
		9573	9631	9655	9975	10137	10141	10210	10246
		10523	10589	10820	10988	11022	11155		
i#restore_traps	5617	5631							
id	4739	6429	6433/M	6447	6490	6490/M	6494	6502	7450
		7450/M	7454	7466	7501	7501/M	7512	7529	7529/M
		7547	7573	7573/M	7599	7653	7653/M	7725	7748
		7748/M	7757	7806	7806/M	7927	7956	7956/M	7983
		8017	8017/M	8039	8070	8070/M	8104	8134	8134/M
		8176	8265	8265/M	8270	8282	8282/M	8292	8444
		8444/M	8451	8508	8532	8532/M	8553	8607	8607/M
		8610	8677	8727	8727/M	8756	8783	8783/M	8870
		8897	8897/M	8961	9004	9004/M	9010	9028	9048
		9100	9100/M	9104	9211	9247	9247/M	9250	9257
		9271	9329	9329/M	9339	9469	9469/M	9486	9504
		9504/M	9539	9573	9607	9607/M	9631	9655	9951
		9951/M	9975	10137	10141	10170	10170/M	10210	10233
		10233/M	10246	10461	10461/M	10523	10589	10593	10593/M
		10820	10981	10981/M	10988	11022	11107	11107/M	11155
idle_count	2479	10776/M	10776						
idle_io_active	2476	10768/M	10768						
idle_no_io_active	2475	10771/M	10771						
idle_resume_sys_task_kind	9492	9512							
idle_resume_sys_task_kind	9585	9609							
idle_start_time	2477	10735/M	10766						
idle_status	4937	6629/M	7837	7888	7891/M	7893	7969	7973/M	7974
		8162	8165/M	8167	8217	8252/M	8445/M	8533	8608/M
		8614	8615/M	8753	8754	8754	9298/M	9394/M	9399
		9409/M	9420/M	9422/M	9472	9482/M	9526/M	9531/M	9533/M
		9560/M	9565/M	9567/M	9616	9626/M	9640	9650/M	9883
		9883	10080	10527	10528/M	10530	10530	10550	10550/M
		10569	10575	10582/M	10907	10908/M	11007/M	11132	
idle_time	10432	10766/M	10769	10772	10774/P				
idle_type	2478	10734	10737/M	10739/M	10765	10767	10775/M		
ijl_bi	7647	7693	7694/S	7696					
ijl_bn	7648	7691	7692/S	7694/S	7697				
ijl_ordinal	1172	8576/P	8608/P	8617/P	8619	8739/P	8754/P	8917/P	9717/P
		9894/P	10466/P	10530/P	10583/P	10635/M	10636/P	10678/M	10679/P
ijl_ordinal	2247	8464/M							
ijl_ordinal	4689	8438	9458/P	9461/P	9596/P				
ijl_ordinal	4829	7661/M	7675/M	7688	7691	7696/M	7697/M	7698/P	
ijl_ordinal	4889	7816/M	9969/P	9997/P	10005/P	10020/P	10033/P	10047/P	10064/P
		10071/P							
ijl_ordinal	4934	6562	6624/M	6966/P	7054	7054	7054	7054	7056

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7056	7056	7056	7058	7058	7105	7105	7148
		7148	7217	7217	7257	7257	7319	7319	7458/P
		7459/P	7503/P	7505/P	7537/P	7539/P	7575/P	7577/P	7809/P
		7816	7866/P	7866	7866	7866	7866	7866	7866
		7866	7866	7866	7866	7892/P	7898/P	7898	7898
		7898	7898	7898	7898	7898	7898	7898	7898
		7910/P	7959/P	7972/P	7978/P	7978	7979	7979	7979
		7979	7979	7979	7979	7979	7979	8140/P	8166/P
		8172/P	8172	8172	8172	8172	8172	8172	8172
		8172	8172	8172	8227/P	8227	8227	8227	8227
		8227	8227	8227	8227	8227	8227	8253/P	8445/M
		8499/P	8499	8499	8499	8499	8499	8499	8499
		8499	8499	8499	8608/M	8630/P	8630	8630	8630
		8630	8630	8630	8630	8630	8630	8630	8837/P
		8839/P	9196/P	9406/P	9406	9406	9406	9406	9406
		9406	9406	9406	9406	9406	9479/P	9479	9479
		9479	9479	9479	9479	9479	9479	9479	9479
		9623/P	9623	9623	9623	9623	9623	9623	9623
		9623	9623	9623	9647/P	9647	9647	9647	9647
		9647	9647	9647	9647	9647	9647	10201/P	10201
		10201	10201	10201	10201	10201	10201	10201	10201
		10201	10512/P	10519/P	10519	10519	10519	10519	10519
		10519	10519	10519	10519	10519	10550/P	10550/P	10550/P
		10550	10550	10550	10550	10550	10550	10550	10550
		10550	10550	10579/P	10579	10579	10579	10579	10579
		10579	10579	10579	10579	10579	10635	10678	10906/P
		10913/P	10925/P	10925	10925	10925	10925	10925	10925
		10925	10925	10925	10925	11001/M	11139/P	11139	11139
		11139	11139	11139	11139	11139	11139	11139	11139
		11146/P	11151/P	11151	11151	11151	11151	11151	11151
		11151	11151	11151	11151				
ijl_ordinal	5385	5389/S	5389/S						
ijl_ordinal	5678	5698/P							
ijl_ordinal	6560	6562/M							
ijl_ordinal	6567	6612/S	6612/S						
ijl_ordinal	6569	6612/P	6624						
ijl_ordinal	6863	6966/S	6966/S						
ijl_ordinal	7416	7458/S	7458/S						
ijl_ordinal	7416	7459/P							
ijl_ordinal	7472	7503/S	7503/S						
ijl_ordinal	7472	7505/P							
ijl_ordinal	7517	7537/S	7537/S						
ijl_ordinal	7517	7539/P							
ijl_ordinal	7562	7575/S	7575/S						
ijl_ordinal	7562	7577/P							
ijl_ordinal	7618	7665/S	7665/S						
ijl_ordinal	7618	7668/P	7698/P						
ijl_ordinal	7621	7668/P	7675						
ijl_ordinal	7766	7809/S	7809/S	7866/S	7866/S	7898/S	7898/S		
ijl_ordinal	7932	7959/S	7959/S	7978/S	7979/S				
ijl_ordinal	8110	8140/S	8140/S	8172/S	8172/S				
ijl_ordinal	8182	8227/S	8227/S						
ijl_ordinal	8381	8439/S	8439/S	8445/S	8445/S	8499/S	8499/S		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	8381	8445/P	8445						
ijl_ordinal	8413	8438/M	8439/P	8445/P	8464				
ijl_ordinal	8581	8608/P	8608						
ijl_ordinal	8581	8608/S	8608/S	8630/S	8630/S				
ijl_ordinal	8770	8837/S	8837/S						
ijl_ordinal	9057	9196/S	9196/S						
ijl_ordinal	9349	9406/S	9406/S						
ijl_ordinal	9446	9458/S	9458/S	9479/S	9479/S				
ijl_ordinal	9584	9596/S	9596/S	9623/S	9623/S	9647/S	9647/S		
ijl_ordinal	9936	9969/S	9969/S						
ijl_ordinal	10157	10201/S	10201/S						
ijl_ordinal	10159	10171/P							
ijl_ordinal	10226	10234/P							
ijl_ordinal	10364	10393/P							
ijl_ordinal	10408	10512/S	10512/S	10519/S	10519/S	10550/S	10550/S	10550/S	10550/S
		10579/S	10579/S	10636/S	10636/S	10679/S	10679/S		
ijl_ordinal	10879	10906/S	10906/S	10925/S	10925/S				
ijl_ordinal	10964	11026/S	11026/S						
ijl_ordinal	11095	11139/S	11139/S	11146/S	11146/S	11151/S	11151/S		
ijl_p	6863	7054	7054						
ijl_p	6863	7056	7056						
ijl_p	7071	7106	7149						
ijl_p	7183	7218	7258						
ijl_p	7766	7866	7866	7898	7898				
ijl_p	7766	7866	7866	7898	7898				
ijl_p	7932	7979	7979						
ijl_p	7932	7979	7979						
ijl_p	8110	8172	8172						
ijl_p	8110	8172	8172						
ijl_p	8182	8227	8227						
ijl_p	8182	8227	8227						
ijl_p	8381	8499	8499						
ijl_p	8381	8499	8499						
ijl_p	8581	8630	8630						
ijl_p	8581	8630	8630						
ijl_p	9349	9406	9406						
ijl_p	9349	9406	9406						
ijl_p	9446	9479	9479						
ijl_p	9446	9479	9479						
ijl_p	9584	9623	9623	9647	9647				
ijl_p	9584	9623	9623	9647	9647				
ijl_p	10157	10201	10201						
ijl_p	10157	10201	10201						
ijl_p	10408	10519	10519	10550	10550	10579	10579		
ijl_p	10408	10519	10519	10550	10550	10579	10579		
ijl_p	10879	10925	10925						
ijl_p	10879	10925	10925						
ijl_p	11095	11139	11139	11151	11151				
ijl_p	11095	11139	11139	11151	11151				
ijl_thread	4944	7369/M	7369	7370/M	7407	7410/M	7410	7411/M	7723
		8536	8635/M	8635	8635/M	8742	8742/M	8742	8742/M
		8754	9264	9299	9410	9429	9483	9536	9570
		9628	9652	9884	10530	11015/M	11015	11016/M	11077

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11085/M							
ijle_p	1173	7369/S	7370/S	7399	8543/M	8543	8549/S	8550/S	8571/P
		8575/P	8576/P	8617/P	8620/M	8624/S	8625/S	8627	8628
		8630/P	8635/S	8635/S	8642/M	8642	8643/M	8643	8644/M
		8675/P	8728/M	8729/M	8729	8730/M	8730	8731/M	8739/M
		8739	8739/M	8739	8739	8739	8739/P	8739/M	8739
		8739	8739	8740/M	8740	8742	8754	8916	8917/P
		8925/M	9139	9140	9144/P	9145/M	9198/S	9199/M	9202
		9203/M	9206/M	9685/M	9685	9687/M	9687	9712	9716
		9717/P	9723/M	9723	9742	9742	9882	10081/P	10466/M
		10466	10466/M	10466	10466	10466	10466/P	10466/M	10466
		10466	10466	10496/S	10497/S	10530	10539/M	10539	10559/M
		10559	10565/M	10565	10567	10568	10578	10578	10579/P
		10583/P	10636/P	10637	10641	10679/P	10680	10686	10693
		10726/M	10726	11110/S	11111/S	11118/M	11118	11122/M	11122
		11128/M	11128	11130	11131	11139/P			
ijle_p	2246	8463/M							
ijle_p	4690	7666	9231	9291	9331	9366	9372	9503	
ijle_p	4830	7665/P	7676/M	7685/P	7703/M	7716/P			
ijle_p	5235	5252							
ijle_p	5286	5293							
ijle_p	5287	5293/P							
ijle_p	5358	5364	5369/M	5373/P	5377/P				
ijle_p	5386	5389/M							
ijle_p	5488	5493	5495/P						
ijle_p	5676	5698	5698/P						
ijle_p	5677	5693	5694	5698/P					
ijle_p	5708	5716	5718/P						
ijle_p	6567	6612/M							
ijle_p	6593	6612/P	6613						
ijle_p	6863	6966/M							
ijle_p	6863	7058							
ijle_p	6865	6884/S	6885/S	6992/M	6992	7051	7051	7054/P	7056/P
		7058/P							
ijle_p	7292	7320							
ijle_p	7416	7458/M							
ijle_p	7416	7459	7459	7459/P					
ijle_p	7416	7459	7459/P						
ijle_p	7419	7458/P	7459/P	7463					
ijle_p	7472	7503/M							
ijle_p	7472	7505	7505	7505/P					
ijle_p	7472	7505	7505/P						
ijle_p	7475	7503/P	7505/P	7509					
ijle_p	7517	7537/M							
ijle_p	7517	7539	7539	7539/P					
ijle_p	7517	7539	7539/P						
ijle_p	7520	7535/M	7537/P	7539/P	7543				
ijle_p	7562	7575/M							
ijle_p	7562	7577	7577	7577/P					
ijle_p	7562	7577	7577/P						
ijle_p	7562	7598	7598/P						
ijle_p	7569	7575/P	7577/P	7581	7598/P				
ijle_p	7618	7665/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijle_p	7618	7668	7668	7668/P	7698	7698	7698/P		
ijle_p	7618	7668	7668/P	7698	7698/P				
ijle_p	7618	7685	7685/P	7716	7716/P				
ijle_p	7620	7668/P	7673	7674	7676				
ijle_p	7766	7809/M	7866/M	7898/M					
ijle_p	7766	7866/S	7866/S	7866/M	7866	7866	7866	7866/P	7866/P
		7866/P	7898/S	7898/S	7898/M	7898	7898	7898	7898/P
		7898/P	7898/P						
ijle_p	7766	7866	7898						
ijle_p	7798	7809/P	7810	7812	7817	7819	7822	7823	7824/S
		7827/M	7842	7843/M	7844/P	7865	7865	7866/P	7879/M
		7879	7884/M	7884	7889/M	7889	7892/P	7894/M	7894
		7896/S	7897	7897	7898/P	7910/P	7914/P	7926/M	
ijle_p	7932	7959/M	7979/M						
ijle_p	7932	7979/S	7979/S	7979/M	7979	7979	7979	7979/P	7979/P
		7979/P							
ijle_p	7932	7979							
ijle_p	7951	7959/P	7970/M	7970	7972/P	7975/M	7975	7977	7977
		7978	7979/P						
ijle_p	8110	8140/M	8172/M						
ijle_p	8110	8172/S	8172/S	8172/M	8172	8172	8172	8172/P	8172/P
		8172/P							
ijle_p	8110	8172							
ijle_p	8129	8140/P	8163/M	8163	8166/P	8168/M	8168	8170	8171
		8171	8172/P						
ijle_p	8182	8222/P	8232/P						
ijle_p	8182	8222	8232						
ijle_p	8182	8227/S	8227/S	8227/M	8227	8227	8227	8227/P	8227/P
		8227/P							
ijle_p	8182	8227/M							
ijle_p	8182	8227							
ijle_p	8184	8218	8222/P	8224/M	8224	8226/S	8227/P	8232/P	8250/M
		8250	8253/P						
ijle_p	8381	8439/M	8445/M	8499/M					
ijle_p	8381	8445/P	8445						
ijle_p	8381	8471	8471/M	8471/P	8471/P				
ijle_p	8381	8499/S	8499/S	8499/M	8499	8499	8499	8499/P	8499/P
		8499/P							
ijle_p	8381	8499							
ijle_p	8414	8439/P	8441/M	8449/P	8463	8465/M	8466/M	8467/M	8468/M
		8469/M	8470/M	8471/P	8473	8496/S	8497/S	8498	8498
		8499/P	8504/S	8505/S					
ijle_p	8581	8608/P	8608						
ijle_p	8581	8608/M	8630/M						
ijle_p	8581	8630/S	8630/S	8630/M	8630	8630	8630	8630/P	8630/P
		8630/P							
ijle_p	8581	8630							
ijle_p	8770	8837/M							
ijle_p	8776	8837/P	8838	8839/P	8848/M				
ijle_p	9057	9196/M							
ijle_p	9221	9231/M	9232	9237	9238/P	9239/M	9249		
ijle_p	9285	9291/M	9295						
ijle_p	9326	9331/M	9332						

*** 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	9349	9406/S	9406/S	9406/M	9406	9406	9406	9406/P	9406/P
		9406/P							
ijle_p	9349	9406/M							
ijle_p	9349	9406							
ijle_p	9357	9366/M	9380	9397	9404	9405	9406/P		
ijle_p	9446	9458/M	9479/M						
ijle_p	9446	9479/S	9479/S	9479/M	9479	9479	9479	9479/P	9479/P
		9479/P							
ijle_p	9446	9479							
ijle_p	9452	9458/P	9460	9461/P	9464	9478	9478	9479/P	
ijle_p	9499	9503/M	9508						
ijle_p	9584	9596/M	9623/M	9647/M					
ijle_p	9584	9623/S	9623/S	9623/M	9623	9623	9623	9623/P	9623/P
		9623/P	9647/S	9647/S	9647/M	9647	9647	9647	9647/P
		9647/P	9647/P						
ijle_p	9584	9623	9647						
ijle_p	9593	9596/P	9598	9599/P	9602	9622	9622	9623/P	9646
		9646	9647/P						
ijle_p	9936	9969/M							
ijle_p	9936	10081	10081/M	10081/P	10081/P				
ijle_p	9945	9869/P	9870	9974	9994/M	9996/P	9997/P	10000/M	10002/M
		10004/M	10005/P	10008	10009	10010/M	10010	10017/M	10019/P
		10020/P	10027/M	10028/M	10029/M	10030/M	10031/M	10032/M	10033/P
		10041/M	10043/M	10044/M	10045/S	10046/P	10047/P	10056/S	10063/P
		10064/P	10066/M	10068/M	10070/P	10071/P			
ijle_p	10157	10201/S	10201/S	10201/M	10201	10201	10201	10201/P	10201/P
		10201/P							
ijle_p	10157	10201/M							
ijle_p	10157	10201							
ijle_p	10158	10171/P	10173/M	10173	10179	10182	10193/S	10194/S	10200
		10200	10201/P						
ijle_p	10225	10234/P	10236/M	10236					
ijle_p	10263	10271	10274	10274	10275	10277/S			
ijle_p	10311	10321	10321	10323	10324/M	10324	10327	10328	10329/S
		10330/S	10332/M	10332	10335/M	10336/S	10337/S	10343/M	10343
ijle_p	10363	10373/S	10379	10389/M	10391/M	10392/P	10393/P		
ijle_p	10408	10512/M	10519/M	10550/M	10550/M	10579/M	10636/M	10679/M	
ijle_p	10408	10519/S	10519/S	10519/M	10519	10519	10519	10519/P	10519/P
		10519/P	10550/S	10550/S	10550/M	10550	10550	10550	10550/P
		10550/P	10550/P	10579/S	10579/S	10579/M	10579	10579	10579
		10579/P	10579/P	10579/P					
ijle_p	10408	10519	10550	10579					
ijle_p	10408	10550/P	10550/M	10550	10550/P	10550/M	10550/M	10550	10550/S
		10550	10550	10550	10550/P	10550/M	10550		
ijle_p	10879	10906/M	10925/M						
ijle_p	10879	10925/S	10925/S	10925/M	10925	10925	10925	10925/P	10925/P
		10925/P							
ijle_p	10879	10925							
ijle_p	10888	10906/P	10910/M	10910	10913/P	10914/M	10917/M	10917	10918/S
		10921	10924	10924	10925/P	10927/M	10927		
ijle_p	10964	11026/M							
ijle_p	10974	11026/P	11031/M	11032/M	11033/M	11034/M	11035/P	11045	11051/P
ijle_p	11069	11075							

*** 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	11095	11139/S	11139/S	11139/M	11139	11139	11139	11139/P	11139/P
		11139/P	11151/S	11151/S	11151/M	11151	11151	11151	11151/P
		11151/P	11151/P						
ijle_p	11095	11139/M	11146/M	11151/M					
ijle_p	11095	11139	11151						
ijlo	4990	8427/P							
ijlo	5676	5698/P							
ijlo	5709	5718/P							
ijlo	7416	7459/P							
ijlo	7472	7505/P							
ijlo	7517	7539/P							
ijlo	7562	7577/P							
ijlo	7618	7668/P	7698/P						
ijlo	10945	10993/S	10993/S	11001	11026/P	11035/P			
in_use	2095	8231							
in_use	4688	5494	5497/M	5497	5698/M	5698	5721/M	5721	7459/M
		7459	7505/M	7505	7539/M	7539	7577/M	7577	7598
		7598/M	7598	7663/M	7663	7668/M	7668	7685	7685/M
		7685	7698/M	7698	7716	7716/M	7716		
in_use_incremented	4831	7662/M	7669/M	7686/M	7699/M	7715	7717/M		
index	102	6491	6492/S	6493/S	6530	6531/S	6532/S	6549	6550/S
		6551/S	6562/S	6634/M	7369/S	7369/S	7370/S	7370	7399
		7403	7452	7452/S	7452/S	7458/S	7459/S	7463/S	7502
		7502/S	7502/S	7503/S	7505/S	7509/S	7531	7532/S	7533/S
		7537/S	7539/S	7543/S	7574	7574/S	7574/S	7575/S	7577/S
		7581/S	7587/S	7666	7674	7701	7749/S	7756/S	7807
		7807/S	7807/S	7808/S	7864/P	7866/P	7898/P	7957	7957/S
		7957/S	7958/S	7979/P	8018/S	8019/S	8022/S	8023/S	8068/S
		8072	8072/S	8072/S	8075	8083	8084	8086	8088
		8135/M	8136/S	8138	8138/S	8138/S	8172/P	8227/P	8235
		8266	8266/S	8266/S	8267/S	8268/M	8445/M	8499/P	8533/S
		8536/S	8544/P	8608/M	8614/S	8615/S	8616/S	8630/P	8635/S
		8635/S	8635/S	8635	8741/P	8742	8742	8753/S	8754
		8803	8837/S	8839/S	8842/S	8844/S	8896	8919/S	8921/S
		9018/S	9102	9102/S	9102/S	9134/S	9189/S	9192/S	9196/S
		9232	9295	9380	9397	9464	9508	9602	9882
		10080/S	10130/S	10132/S	10133/S	10134/S	10191	10472	10530
		10746/M	10783/M	10984/S	10994	10996	11018/M	11075	11116
index_p	4076	5389	6612	6966	7458	7503	7537	7575	7665
		7692	7694	7809	7866	7898	7959	7979	8140
		8172	8227	8439	8445	8499	8608	8630	8837
		9196	9406	9458	9479	9596	9623	9647	9969
		10201	10512	10519	10550	10550	10579	10636	10679
		10906	10925	10993	11026	11139	11146	11151	
inhibit_access	5679	5695/M	5697/M						
inhibit_access	7416	7459/M	7459/M						
inhibit_access	7447	7459/P	7460						
inhibit_access	7472	7505/M	7505/M						
inhibit_access	7499	7505/P	7506						
inhibit_access	7517	7539/M	7539/M						
inhibit_access	7527	7539/P	7540						
inhibit_access	7562	7577/M	7577/M						
inhibit_access	7570	7577/P	7578						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
inhibit_access	7618	7668/M	7668/M	7698/M	7698/M				
inhibit_access	7649	7668/P	7669	7670	7698/P	7699	7700		
initiate_swap_if_possible	9874	8754	9897	10530					
insert_ijl	7352	7372	8635						
insert_ijle_p	6875	6966/P	6967/M	6967					
insert_ijle_p	7766	7866/P	7866/M	7866	7898/P	7898/M	7898		
insert_ijle_p	7932	7979/P	7979/M	7979					
insert_ijle_p	8110	8172/P	8172/M	8172					
insert_ijle_p	8182	8227/P	8227/M	8227					
insert_ijle_p	8381	8499/P	8499/M	8499					
insert_ijle_p	8581	8630/P	8630/M	8630					
insert_ijle_p	9349	9406/P	9406/M	9406					
insert_ijle_p	9446	9479/P	9479/M	9479					
insert_ijle_p	9584	9623/P	9623/M	9623	9647/P	9647/M	9647		
insert_ijle_p	10157	10201/P	10201/M	10201					
insert_ijle_p	10408	10519/P	10519/M	10519	10550/P	10550/M	10550	10579/P	10579/M
		10579							
insert_ijle_p	10879	10925/P	10925/M	10925					
insert_ijle_p	11095	11139/P	11139/M	11139	11151/P	11151/M	11151		
insert_ptle_p	6874	6965/M	6966/P	6968/M	6969/S	6971/M	6978	6981/M	6984/S
insert_ptle_p	7766	7866/M	7866/P	7866/M	7866/S	7866/M	7866	7866/M	7866/S
		7898/M	7898/P	7898/M	7898/S	7898/M	7898	7898/M	7898/S
insert_ptle_p	7932	7979/M	7979/P	7979/M	7979/S	7979/M	7979	7979/M	7979/S
insert_ptle_p	8110	8172/M	8172/P	8172/M	8172/S	8172/M	8172	8172/M	8172/S
insert_ptle_p	8182	8227/M	8227/P	8227/M	8227/S	8227/M	8227	8227/M	8227/S
insert_ptle_p	8381	8499/M	8499/P	8499/M	8499/S	8499/M	8499	8499/M	8499/S
insert_ptle_p	8581	8630/M	8630/P	8630/M	8630/S	8630/M	8630	8630/M	8630/S
insert_ptle_p	9349	9406/M	9406/P	9406/M	9406/S	9406/M	9406	9406/M	9406/S
insert_ptle_p	9446	9479/M	9479/P	9479/M	9479/S	9479/M	9479	9479/M	9479/S
insert_ptle_p	9584	9623/M	9623/P	9623/M	9623/S	9623/M	9623	9623/M	9623/S
		9647/M	9647/P	9647/M	9647/S	9647/M	9647	9647/M	9647/S
insert_ptle_p	10157	10201/M	10201/P	10201/M	10201/S	10201/M	10201	10201/M	10201/S
insert_ptle_p	10408	10519/M	10519/P	10519/M	10519/S	10519/M	10519	10519/M	10519/S
		10550/M	10550/P	10550/M	10550/S	10550/M	10550	10550/M	10550/S
		10579/M	10579/P	10579/M	10579/S	10579/M	10579	10579/M	10579/S
insert_ptle_p	10879	10925/M	10925/P	10925/M	10925/S	10925/M	10925	10925/M	10925/S
insert_ptle_p	11095	11139/M	11139/P	11139/M	11139/S	11139/M	11139	11139/M	11139/S
		11151/M	11151/P	11151/M	11151/S	11151/M	11151	11151/M	11151/S
insert_ptlo	6863	7054/M	7054/S	7054/S	7054/M	7054	7054	7054/S	7054
		7054	7054/M	7054/M	7054	7054	7054/S	7054/S	
insert_ptlo	6863	7056/M	7056/S	7056/S	7056/M	7056	7056	7056/S	7056
		7056/M	7056/M	7056	7056/S	7056/S			
insert_ptlo	6863	7058/M	7058/S	7058/M	7058				
insert_ptlo	7081	7092/M	7114/S	7115/S	7116/M	7126	7131	7133/S	7134
		7136	7145/M	7152/M	7158	7165	7169/S	7170/S	
insert_ptlo	7193	7204/M	7229/S	7230/S	7231/M	7241	7242	7243/S	7244
		7254/M	7262/M	7267	7278/S	7279/S			
insert_ptlo	7301	7314/M	7336/S	7337/M	7347				
insert_ptlo	7766	7866/M	7866/S	7866/S	7866/M	7866	7866	7866/S	7866
		7866	7866/M	7866/M	7866	7866	7866/S	7866/S	7898/M
		7898/S	7898/S	7898/M	7898	7898	7898/S	7898	7898
		7898/M	7898/M	7898	7898	7898/S	7898/S		
insert_ptlo	7766	7866/M	7866/S	7866/S	7866/M	7866	7866	7866/S	7866

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7866/M	7866/M	7866	7866/S	7866/S	7898/M	7898/S	7898/S
		7898/M	7898	7898	7898/S	7898	7898/M	7898/M	7898
		7898/S	7898/S						
insert_ptlo	7766	7866/M	7866/S	7866/M	7866	7898/M	7898/S	7898/M	7898
insert_ptlo	7932	7979/M	7979/S	7979/S	7979/M	7979	7979	7979/S	7979
		7979	7979/M	7979/M	7979	7979	7979/S	7979/S	
insert_ptlo	7932	7979/M	7979/S	7979/S	7979/M	7979	7979	7979/S	7979
		7979/M	7979/M	7979	7979/S	7979/S			
insert_ptlo	7932	7979/M	7979/S	7979/M	7979				
insert_ptlo	8110	8172/M	8172/S	8172/S	8172/M	8172	8172	8172/S	8172
		8172	8172/M	8172/M	8172	8172	8172/S	8172/S	
insert_ptlo	8110	8172/M	8172/S	8172/S	8172/M	8172	8172	8172/S	8172
		8172/M	8172/M	8172	8172/S	8172/S			
insert_ptlo	8110	8172/M	8172/S	8172/M	8172				
insert_ptlo	8182	8227/M	8227/S	8227/S	8227/M	8227	8227	8227/S	8227
		8227	8227/M	8227/M	8227	8227	8227/S	8227/S	
insert_ptlo	8182	8227/M	8227/S	8227/S	8227/M	8227	8227	8227/S	8227
		8227/M	8227/M	8227	8227/S	8227/S			
insert_ptlo	8182	8227/M	8227/S	8227/M	8227				
insert_ptlo	8381	8499/M	8499/S	8499/S	8499/M	8499	8499	8499/S	8499
		8499	8499/M	8499/M	8499	8499	8499/S	8499/S	
insert_ptlo	8381	8499/M	8499/S	8499/S	8499/M	8499	8499	8499/S	8499
		8499/M	8499/M	8499	8499/S	8499/S			
insert_ptlo	8381	8499/M	8499/S	8499/M	8499				
insert_ptlo	8581	8630/M	8630/S	8630/S	8630/M	8630	8630	8630/S	8630
		8630	8630/M	8630/M	8630	8630	8630/S	8630/S	
insert_ptlo	8581	8630/M	8630/S	8630/S	8630/M	8630	8630	8630/S	8630
		8630/M	8630/M	8630	8630/S	8630/S			
insert_ptlo	8581	8630/M	8630/S	8630/M	8630				
insert_ptlo	9349	9406/M	9406/S	9406/S	9406/M	9406	9406	9406/S	9406
		9406	9406/M	9406/M	9406	9406	9406/S	9406/S	
insert_ptlo	9349	9406/M	9406/S	9406/S	9406/M	9406	9406	9406/S	9406
		9406/M	9406/M	9406	9406/S	9406/S			
insert_ptlo	9349	9406/M	9406/S	9406/M	9406				
insert_ptlo	9446	9479/M	9479/S	9479/S	9479/M	9479	9479	9479/S	9479
		9479	9479/M	9479/M	9479	9479	9479/S	9479/S	
insert_ptlo	9446	9479/M	9479/S	9479/S	9479/M	9479	9479	9479/S	9479
		9479/M	9479/M	9479	9479/S	9479/S			
insert_ptlo	9446	9479/M	9479/S	9479/M	9479				
insert_ptlo	9584	9623/M	9623/S	9623/S	9623/M	9623	9623	9623/S	9623
		9623	9623/M	9623/M	9623	9623	9623/S	9623/S	9647/M
		9647/S	9647/S	9647/M	9647	9647	9647/S	9647	9647
		9647/M	9647/M	9647	9647	9647/S	9647/S		
insert_ptlo	9584	9623/M	9623/S	9623/S	9623/M	9623	9623	9623/S	9623
		9623/M	9623/M	9623	9623/S	9623/S	9647/M	9647/S	9647/S
		9647/M	9647	9647	9647/S	9647	9647/M	9647/M	9647
		9647/S	9647/S						
insert_ptlo	9584	9623/M	9623/S	9623/M	9623	9647/M	9647/S	9647/M	9647
insert_ptlo	10157	10201/M	10201/S	10201/S	10201/M	10201	10201	10201/S	10201
		10201	10201/M	10201/M	10201	10201	10201/S	10201/S	
insert_ptlo	10157	10201/M	10201/S	10201/S	10201/M	10201	10201	10201/S	10201
		10201/M	10201/M	10201	10201/S	10201/S			
insert_ptlo	10157	10201/M	10201/S	10201/M	10201				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
insert_ptlo	10408	10519/M	10519/S	10519/S	10519/M	10519	10519	10519/S	10519
		10519	10519/M	10519/M	10519	10519	10519/S	10519/S	10550/M
		10550/S	10550/S	10550/M	10550	10550	10550/S	10550	10550
		10550/M	10550/M	10550	10550	10550/S	10550/S	10579/M	10579/S
		10579/S	10579/M	10579	10579	10579/S	10579	10579	10579/M
		10579/M	10579	10579	10579/S	10579/S			
insert_ptlo	10408	10519/M	10519/S	10519/S	10519/M	10519	10519	10519/S	10519
		10519/M	10519/M	10519	10519/S	10519/S	10550/M	10550/S	10550/S
		10550/M	10550	10550	10550/S	10550	10550/M	10550/M	10550
		10550/S	10550/S	10579/M	10579/S	10579/S	10579/M	10579	10579
		10579/S	10579	10579/M	10579/M	10579	10579/S	10579/S	
insert_ptlo	10408	10519/M	10519/S	10519/M	10519	10550/M	10550/S	10550/M	10550
		10579/M	10579/S	10579/M	10579				
insert_ptlo	10879	10925/M	10925/S	10925/S	10925/M	10925	10925	10925/S	10925
		10925	10925/M	10925/M	10925	10925	10925/S	10925/S	
insert_ptlo	10879	10925/M	10925/S	10925/S	10925/M	10925	10925	10925/S	10925
		10925/M	10925/M	10925	10925/S	10925/S			
insert_ptlo	10879	10925/M	10925/S	10925/M	10925				
insert_ptlo	11095	11139/M	11139/S	11139/S	11139/M	11139	11139	11139/S	11139
		11139	11139/M	11139/M	11139	11139	11139/S	11139/S	11151/M
		11151/S	11151/S	11151/M	11151	11151	11151/S	11151	11151
		11151/M	11151/M	11151	11151	11151/S	11151/S		
insert_ptlo	11095	11139/M	11139/S	11139/S	11139/M	11139	11139	11139/S	11139
		11139/M	11139/M	11139	11139/S	11139/S	11151/M	11151/S	11151/S
		11151/M	11151	11151	11151/S	11151	11151/M	11151/M	11151
		11151/S	11151/S						
insert_ptlo	11095	11139/M	11139/S	11139/M	11139	11151/M	11151/S	11151/M	11151
integer_dp_sets	10430	10666	10668	10717	10719	10724			
interactive_task_gtid	1310	7812	7827/M	10041/M					
iot$io_error	2175	1331	2141						
iot$transfer_count	3031	3019							
jcb_identifier	2232	8460/M							
jcb_p	1155	8739	8739	8739/P	8739/M	8744/M	9744	9744	9745/P
		9746/M	10466	10466	10466/P	10466/M	10470/M	10471/M	10638/M
		10648	10681/M	10694	11114/M	11115/M			
jcb_p	8418	8459/M	8460/M	8461/M	8462/M	8463/M	8464/M		
jcb_p	8682	8754/M	8754	8754/M	8754/M				
jcb_p	9284	9302/M	9303	9305/M	9307/M				
jcb_p	9358	9368/M	9435	9437/M	9439/M				
jcb_p	9878	9887/M	9888	9890/M	9892/M				
jcb_p	10408	10530/M	10530	10530/M	10530/M				
jm	5984	8480	8654	8660					
jm	8417	8477/M	8480/M	8483	8488				
jm	8599	8654/M	8660/M	8664/M	8668	8673			
jmc$batch	1695	10271							
jmc$dc_maximum_service_limit	1478	10004	10030	10284					
jmc$detached_job_wait_time_max	2275	2272							
jmc$dp_conversion	682	6693	6977	7013	7864	7866	7866	7898	7898
		7979	7979	8172	8172	8227	8227	8499	8499
		8630	8630	8739	8739	8739	8739	8739	8739
		8739	8739	8739	8739	8739	9406	9406	9417
		9479	9479	9528	9562	9623	9623	9647	9647

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9694	9694	9694	9694	9694	9694	9694	9694
		9694	9694	9694	9768	9774	9780	9782	9790
		9791	9797	9799	9801	9804	9807	9834	10198
		10201	10201	10466	10466	10466	10466	10466	10466
		10466	10466	10466	10466	10466	10519	10519	10550
		10550	10579	10579	10711	10713	10715	10725	10774
		10774	10774	10774	10774	10774	10774	10774	10774
		10774	10774	10925	10925	11139	11139	11151	11151
jmc$dpb_absolute	2531	10177							
jmc$dpb_positive	2531	10178							
jmc$dsw_adjust_cpu_selections	1429	10237							
jmc$dsw_update_job_task_enviro	1428	10174							
jmc$dsw_update_keypoint_masks	1427	9460	9598						
jmc$highest_det_job_wait_time	2285	2275	2286						
jmc$highest_prio_age_interval	2626	2617	2627						
jmc$highest_service_accumulator	1759	1760							
jmc$highest_service_factor_valu	2650	2643							
jmc$highest_working_set_size	2321	2312	2322	2324	2326	2328			
jmc$ies_entry_free	1534	7695							
jmc$ies_job_in_memory	1537	9332	10568	11131					
jmc$ies_job_in_memory_non_swap	1536	8471/P	9372	10081/P					
jmc$ies_job_swapped	1539	1548	9197						
jmc$ies_job_terminating	1535	5694	7459	7505	7539	7577	7668	7698	
jmc$ies_swapin_candidate	1544	7823							
jmc$ies_swapin_in_progress	1538	1547	5371	5372	5375	5376	8471	8471	8471
		8471	10081	10081	10081	10081			
jmc$inhibit_xcb_access	1592	5693	7459	7505	7539	7577	7668	7698	
jmc$iss_executing	1553	8465							
jmc$iss_idle_tasks_initiated	1554	1581							
jmc$iss_swapin_io_complete	1579	1582							
jmc$iss_swapin_requested	1575	1582							
jmc$iss_swapout_complete	1574	1581	11031						
jmc$iss_swapped_io_cannot_init	1565	1592							
jmc$iss_swapped_no_io	1556	1591							
jmc$keyword_offset_maximum	1776	2313	2618						
jmc$kjl_maximum_entries	1211	1204	1205	1711					
jmc$kol_maximum_entries	1221	1206							
jmc$lock_ajl	5471	5494	5495/P	5497	5698/P	5698	5718/P	5721	7459/P
		7459	7505/P	7505	7539/P	7539	7577/P	7577	7598
		7598/P	7598	7664	7668/P	7668	7685	7685/P	7685
		7698/P	7698	7716	7716/P	7716			
jmc$max_active_jobs	1202	2599	2607	2608					
jmc$max_ajl_ord	'203	1196	1202	4699	4759				
jmc$max_completed_job_count	6246	6239							
jmc$max_dispatching_control	1489	1493	10280	10386					
jmc$max_dispatching_priority	684	644	647	648	4780	10185	10186		
jmc$max_ijl_index_count	1251	4074							
jmc$maximum_job_classes	1689	1692							
jmc$maximum_job_count	1218	1211	6224						
jmc$maximum_output_count	1228	1221							
jmc$maximum_service_classes	1792	1795							
jmc$min_dispatching_control	1488	1492	10027	10043	10281	10381	10400		
jmc$min_dispatching_priority	685	4780	10187	10188					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$min_ecc	4126	4127	4134						
jmc$min_ecc_sch	4134	4135	4137	4139	4141	4143	4145	4147	4149
		4151	4153	4155	4157	4159	4161	4163	4165
		4167	4169	4174	4178	4181	4184	4187	4190
		4193	4196	4199	4202	4206	4209	4213	4216
		4219	4222	4225	4228	4231	4235	4238	4242
		4245	4248	4252	4255	4259	4263	4267	4271
		4274	4277	4280					
jmc$null_ajl_ordinal	4699	5698	5717	7459	7505	7539	7577	7668	7698
		8450							
jmc$null_dispatching_priority	686	6602/S	6608/S	6649	6651/S	6652/S	6654/S	6656/S	6752/S
		6776/S	8445/S	8445/S	8544	8544/S	8544/S	8544/S	8544/S
		8608/S	8608/S	8739	8741	8741/S	8741/S	8741/S	8741/S
		9694	9773	9991	9994	10031	10032	10113/S	10114/S
		10116/S	10118/S	10323	10327	10466	10774/P	10774	10984
		10997/S	10997/S	11086	11086/S	11086/S	11086/S	11086/S	
jmc$null_service_class	1785	1786							
jmc$prior_subsystem_tbls_locked	637	6919	6920	7866	7866	7898	7898	7979	7979
		8172	8172	8227	8227	8499	8499	8630	8630
		9406	9406	9479	9479	9623	9623	9647	9647
		10201	10201	10488	10488/S	10519	10519	10550	10550
		10579	10579	10925	10925	11139	11139	11151	11151
jmc$prior_system_tbls_locked	636	6929	6930	7866	7866	7898	7898	7979	7979
		8172	8172	8227	8227	8499	8499	8630	8630
		9406	9406	9479	9479	9623	9623	9647	9647
		10201	10201	10483	10484/S	10519	10519	10550	10550
		10579	10579	10925	10925	11139	11139	11151	11151
jmc$priority_aging_interval_max	2617	2614							
jmc$priority_p1	698	645	3786	8739	9694	9767	10466	10774	
jmc$priority_p10	707	629	633	636	646				
jmc$priority_p11	708	630	632						
jmc$priority_p12	709	631							
jmc$priority_p13	710	634							
jmc$priority_p14	711	646	3786						
jmc$priority_p8	705	645	8739	9694	9767	10466	10774		
jmc$priority_p9	706	637	8739	9694	9760	10466	10774		
jmc$priority_system_job	629	10028	10029						
jmc$required_offset	1774	2327							
jmc$reserved_ajls	1207	1202							
jmc$service_accumulator_maximum	1751	1748							
jmc$service_factor_value_max	2643	2640							
jmc$sr_idle_dispatching	1805	9333/P	9392						
jmc$sr_long_wait	1803	9310/P							
jmc$subsystem_priority_change	5426	7825/P							
jmc$swapping_ajl	5470	8427/P	8449/P	8575/P					
jmc$system_default_offset	1775	1776	2329						
jmc$system_supplied_name_size	2016	2013							
jmc$unlimited_offset	1772	1761	2276	2287	2323	2628			
jmc$unspecified_offset	1773	2325							
jmc$working_set_size_maximum	2312	2309							
jme$non_existent_job	4135	9976/P							
jmp$assign_ajl_entry	5460	8427							
jmp$assign_ajl_with_lock	5730	5698	5718	7459	7505	7539	7577	7668	7698

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmp$change_ijl_entry_status	5357	5380	8471	10081					
jmp$decrement_swapped_job_count	5381	5377	8471	10081					
jmp$free_ajl_entry	5473	8575							
jmp$free_ajl_with_lock	5480	5495	7598	7685	7716	8449			
jmp$get_ijle_p	5385	5391	6612	6966	7458	7503	7537	7575	7665
		7809	7866	7898	7959	7979	8140	8172	8227
		8439	8445	8499	8608	8630	8837	9196	9406
		9458	9479	9596	9623	9647	9969	10201	10512
		10519	10550	10550	10579	10636	10679	10906	10925
		11026	11139	11146	11151				
jmp$increment_swapped_job_count	5396	5373	8471	10081	11051				
jmp$lock_ajl_with_lock	5707	5698	5725	7459	7505	7539	7577	7668	7698
jmp$ready_task_in_swapped_job	5400	7892	7910	7972	8166	8253	8617	10550	10583
		10913							
jmp$set_job_terminated	5406	8576							
jmp$set_scheduler_event	5413	7825							
jmp$set_swapout_candidate	5442	9310							
jmp$subsystem_priority_change	5449	7844							
jmp$swap_non_dispatchable_job	5455	9335							
jmp$unlock_ajl_with_lock	5487	5500	7598	7685	7716				
jmp$update_service_class_stats	5506	8571							
jmt$active_job_list	4695	6356							
jmt$active_job_list_entry	4687	4695							
jmt$ajl_ordinal	1196	1153	1267	4828	4971	4989	5443	5456	5464
		5491	5552	5691	5710	5713	5735	6360	6360
		7609	8416	9218	9279	9323	9350	9447	
jmt$completed_job_count_range	6239	6233							
jmt$delayed_swapin_work	1430	1297	1434	10174	10237				
jmt$detached_job_wait_time	2272	2257							
jmt$dispatching_control	1459	2582	10268	10369					
jmt$dispatching_control_index	1492	1449	1459	4903	10269				
jmt$dispatching_control_info	4902	4891	7799						
jmt$dispatching_controls	1462	1460	9943						
jmt$dispatching_priority	644	1145	1279	1450	1451	1452	1464	2530	2532
		4116	4904	4942	4943	5106	6184	6331	6405
		7768	7797	8777	8892	9093	9755	9758	9827
		10955							
jmt$dispatching_priority_set	4116	4787	4788	4789	4815	4816	4817	4818	6179
		6180	6693	6694	6977	6979	7013	7014	7864
		7864	7866	7866	7866	7866	7898	7898	7898
		7898	7979	7979	7979	7979	8172	8172	8172
		8172	8227	8227	8227	8227	8499	8499	8499
		8499	8630	8630	8630	8630	8739	8739	8739
		8739	8739	8739	8739	8739	8739	8739	8739
		8739	8739	8739	8739	9406	9406	9406	9406
		9417	9417	9479	9479	9479	9479	9528	9528
		9562	9562	9623	9623	9623	9623	9647	9647
		9647	9647	9694	9694	9694	9694	9694	9694
		9694	9694	9694	9694	9694	9694	9694	9694
		9694	9764	9765	9766	9768	9768	9779	9781
		9782	9782	9796	9799	9803	9806	9807	9807
		9834	9836	9860	10198	10198	10201	10201	10201
		10201	10466	10466	10466	10466	10466	10466	10466

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10466	10466	10466	10466	10466	10466	10466	10466
		10519	10519	10519	10519	10550	10550	10550	10550
		10579	10579	10579	10579	10660	10710	10712	10714
		10725	10725	10774	10774	10774	10774	10774	10774
		10774	10774	10774	10774	10774	10774	10774	10774
		10774	10925	10925	10925	10925	11139	11139	11139
		11139	11151	11151	11151	11151			
jmt$idle_dispatch_controls	6184	6181							
jmt$idle_dispatching_controls	6178	6173							
jmt$idle_dispatching_entry	6186	6184							
jmt$ijl_block_index	1247	1243	4076	7647	7693	7708			
jmt$ijl_block_number	1246	1242	4064	4065	7648				
jmt$ijl_dispatching_control	1448	1280							
jmt$ijl_entry_status	1534	1266	5359	5362	6203	6203			
jmt$ijl_entry_status_statistics	6203	6196							
jmt$ijl_ordinal	1241	1172	1286	1314	2031	2032	2094	2133	2247
		4689	4829	4889	4934	4990	5385	5400	5407
		5461	5511	5521	5661	5678	5709	5732	5992
		6308	6361	6361	6560	6569	7621	8413	10159
		10226	10364	10945					
jmt$ijl_p	4062	6357	6357						
jmt$ijl_page_fault_count	1608	1603	1604	1605					
jmt$ijl_page_stats	1602	1598							
jmt$ijl_service_class_stats	1596	1301							
jmt$ijl_statistics	1641	1300							
jmt$ijl_swap_count	1617	1613	1614						
jmt$ijl_swap_counts	1612	1320	1599						
jmt$ijl_swap_status	1552	1269	1270	1271					
jmt$ijle_size	4086	6358	6358						
jmt$initiated_job_list_block	4073	4079							
jmt$initiated_job_list_entry	1263	1173	2058	2246	4076	4690	4830	5235	5287
		5358	5381	5386	5396	5401	5408	5450	5474
		5481	5488	5507	5516	5522	5542	5660	5677
		5708	6593	6865	6875	7071	7183	7292	7419
		7475	7520	7569	7620	7650	7798	7951	8129
		8184	8414	8776	8890	9092	9221	9285	9326
		9357	9452	9499	9593	9945	10158	10225	10263
		10311	10363	10426	10834	10888	10974	11069	11099
jmt$initiated_job_list_p	4079	4063							
jmt$input_file_location	1731	1726							
jmt$job_abort_disposition	1740	1724							
jmt$job_class	1692	1325	6228						
jmt$job_class_count	6229	6228							
jmt$job_class_counts	6228	6218							
jmt$job_control_block	2228	1155	8418	9284	9358	9878			
jmt$job_count_range	6224	6215	6216	6217	6230	6231	6232	6251	6252
jmt$job_counts	6214	6209							
jmt$job_mode	1695	1282							
jmt$job_priority	1700	1322	1323	2591	2592	2593	2594		
jmt$job_recovery_disposition	1743	1725							
jmt$job_scheduler_events	5417	5413	5437	5439					
jmt$job_system_id	2291	2243							
jmt$kjl_index	1711	1268	2291						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$maximum_active_jobs	2599	2576							
jmt$mtr_serv_class_stat_entry	6268	6264							
jmt$priority_aging_interval	2614	2584							
jmt$queue_file_ijl_information	1723	1307							
jmt$sc_cp_stat	6296	6275	6276						
jmt$sc_pf_stat	6297	6280	6281	6282					
jmt$sc_swap_count	6299	6286	6287	6291	6293				
jmt$sc_swap_stat	6298	6288	6289	6290	6292				
jmt$scheduling_data	1313	1291							
jmt$scheduling_priority	2590	2583							
jmt$service_accumulator	1748	1315	1316	1317	2574	2575			
jmt$service_class_attributes	2562	6263	9094						
jmt$service_class_count	6250	6249							
jmt$service_class_counts	6249	6219							
jmt$service_class_cp_time	6274	6269							
jmt$service_class_entry	6262	6258							
jmt$service_class_index	1795	1326	2567	2577	6249	6258	6314	10976	
jmt$service_class_name	2632	2569	2570						
jmt$service_class_page_faults	6279	6270							
jmt$service_class_swap_stats	6285	6271							
jmt$service_factor_value	2640	2578							
jmt$service_factors	2636	2578							
jmt$swap_data	1329	1293							
jmt$swapout_reasons	1798	1321	5444	9351					
jmt$swapped_job_entry	1813	1338	2059	2266					
jmt$system_supplied_name	2013	1264	2241	4890	5991				
jmt$task_time_slice	1502	1482	1483						
jmt$time_slice_values	1481	1466	2543	6879					
jmt$user_dispatching_priority	645	4800							
jmt$user_supplied_name	2295	2242							
jmt$working_set_size	2309	2253	2254						
jmtr_ptlo	10972	10993/M	11014	11015/S	11016/S				
jmtr_taskid	4988	8445/P	8454	8461	8470	8499/P			
jmv$ajl_p	6356	5494	5497/M	5497	5698/M	5698	5721/M	5721	7459/M
		7459	7505/M	7505	7539/M	7539	7577/M	7577	7598
		7598/M	7598	7663/M	7663	7666	7668/M	7668	7685
		7685/M	7685	7698/M	7698	7716	7716/M	7716	7896/M
		8226/M	8438	9231	9291	9331	9366	9372	9458/P
		9461/P	9503	9596/P	10550/M	10918/M			
jmv$idle_dispatching_controls	6173	8739/M	8739	9694/M	9694	9798/M	9798	10466/M	10466
		10774/M	10774						
jmv$ijl_entry_status_statistics	6196	5366/M	5367	8471/M	8471	10081/M	10081		
jmv$ijl_p	6357	5389	6612	6966	7458	7503	7537	7575	7665
		7691	7692	7694	7809	7866	7898	7959	7979
		8140	8172	8227	8439	8445	8499	8608	8630
		8837	9196	9406	9458	9479	9586	9623	9647
		9969	10201	10512	10519	10550	10550	10579	10636
		10679	10906	10925	10993	11026	11139	11146	11151
jmv$job_counts	6209	8504/M	8505	8549/M	8550	11046/M	11046		
jmv$null_ijl_ordinal	6308	8846/P	8923/P	10207/P	10243/P	11033	11034		
jmv$service_classes	6257	6884	7866	7898	7979	8172	8227	8496	8499
		8624	8630	9198	9406	9479	9623	9647	10044
		10056	10193	10201	10277	10329	10336	10373	10496

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10519	10550	10579	10925	11110	11139	11151	
jmv$subsystem_priority_changes	6314	7824/M							
jmv$swap_jobs_in_long_wait	6359	9227	9287						
jmv$system_ajl_ordinal	6360	7660	7663/S	7663/S	9227	9287	9503/S	9515	9549
		9596/S	9612	9636					
jmv$system_ijl_ordinal	6361	7661	7665/P	8619	9599/P				
job_cptime	4870	9929/M							
job_fixed_asid	1276	5698/P	5718/P	7459/P	7505/P	7539/P	7577/P	7668/P	7698/P
job_fixed_contiguous_pages	1302	8467/M							
job_is_good_swap_candidate	4691	7896/M	8226/M	10550/M	10918/M				
job_mode	1282	10271							
job_monitor_id	2245	8461/M	8739/P	9745/P	10466/P				
job_monitor_taskid	1281	7369/S	7370/S	7399	7666	7674	7701	8470/M	8635/S
		8635/S	8742	8754	9232	9295	9380	9397	9464
		9508	9602	9882	10530	10994	11075		
job_scheduler_data	1291	6884/S	7824/S	7866/S	7898/S	7979/S	8172/S	8227/S	8496/S
		8499/S	8504/S	8505/S	8549/S	8551/S	8620/M	8624/S	8630/S
		9198/S	9199/M	9201	9202	9202	9203/M	9203	9205
		9206/M	9206	9406/S	9479/S	9623/S	9647/S	10045/S	10056/S
		10193/S	10201/S	10277/S	10329/S	10336/S	10373/S	10496/S	10519/S
		10550/S	10579/S	10925/S	11045	11110/S	11139/S	11151/S	
job_segnum	8411	8426/M	8427/S	8493/P					
jsc$isqi_null	2035	11032							
jsc$isqi_swapped_io_completed	2036	2038							
jsc$isqi_swapped_io_not_init	2035	2038							
jsc$isqi_swapped_out	2036	11035/P							
jsc$min_ecc	4290	4291							
jsc$min_ecc_js	4291	4294	4297	4300	4303	4306	4309	4312	4315
		4318	4321	4324	4327				
jse$job_executing_non_swappable	4312	9373/P							
jse$unable_to_idle_all_tasks	4303	9412/P	9433/P	9543/P	9577/P				
jsp$idle_tasks_complete	5511	8754	9894	10530					
jsp$long_wait_aging	5516	9144	9238						
jsp$relink_swap_queue	5521	11035							
jst$changed_asid_entry	2081	2072							
jst$ijl_swap_queue_id	2035	2030	5523						
jst$ijl_swap_queue_link	2029	1275							
jst$io_control_information	2043	1294							
jst$swap_file_descriptor	2057	1295							
jst$swapped_page_descriptor	2066	2064							
jst$swapped_page_descriptors	2063	2060							
keypoint_enable	2552	6621/M	8445/M	8608/M	8650/M	8650	8658		
keypoint_mask	3121	8488/M	8673/M						
keypoint_register_enable	2553	6622/M	8445/M	8481/M	8608/M	8655/M	8661/M	8665/M	10809
ktt	8402	8473/M	8474	8475					
last_execution_time	2249	8462/M	10471/M	11115/M					
last_lpid_for_job	2237	8744/M	10470/M	10648	10694	11114/M			
last_lpid_for_task	2512	10469/M	10642	10687	11113/M				
last_ptlo	7393	7398/M	7406/M	7407/S	7410/S				
last_ptlo	8682	8742/M	8742/M	8742/S	8742/S				
length	5601	5608	5611/M	5611					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
length	9936	10106	10106/M	10106					
local_set	6660	6694/M	6694/M	6694	6694	6694	6694/M	6694/M	6694
		6694	6694/M	6694	6694	6694/M	6694	6694	6694
		6694							
local_set	6863	6979/M	6979/M	6979	6979	6979	6979/M	6979/M	6979
		6979	6979/M	6979	6979	6979/M	6979	6979	6979
		6979	7014/M	7014/M	7014	7014	7014	7014/M	7014/M
		7014	7014	7014/M	7014	7014	7014/M	7014	7014
		7014	7014						
local_set	7766	7864/M	7864/M	7864	7864	7864	7864/M	7864/M	7864
		7864	7864/M	7864	7864	7864/M	7864	7864	7864
		7864	7866/M	7866/M	7866	7866	7866	7866/M	7866/M
		7866	7866	7866/M	7866	7866	7866/M	7866	7866
		7866	7866	7866/M	7866/M	7866	7866	7866	7866/M
		7866/M	7866	7866	7866/M	7866	7866	7866/M	7866
		7866	7866	7866	7898/M	7898/M	7898	7898	7898
		7898/M	7898/M	7898	7898	7898/M	7898	7898	7898/M
		7898	7898	7898	7898	7898/M	7898/M	7898	7898
		7898	7898/M	7898/M	7898	7898	7898/M	7898	7898
		7898/M	7898	7898	7898	7898			
local_set	7932	7979/M	7979/M	7979	7979	7979	7979/M	7979/M	7979
		7979	7979/M	7979	7979	7979/M	7979	7979	7979
		7979	7979/M	7979/M	7979	7979	7979	7979/M	7979/M
		7979	7979	7979/M	7979	7979	7979/M	7979	7979
		7979	7979						
local_set	8110	8172/M	8172/M	8172	8172	8172	8172/M	8172/M	8172
		8172	8172/M	8172	8172	8172/M	8172	8172	8172
		8172	8172/M	8172/M	8172	8172	8172	8172/M	8172/M
		8172	8172	8172/M	8172	8172	8172/M	8172	8172
		8172	8172						
local_set	8182	8227/M	8227/M	8227	8227	8227	8227/M	8227/M	8227
		8227	8227/M	8227	8227	8227/M	8227	8227	8227
		8227	8227/M	8227/M	8227	8227	8227	8227/M	8227/M
		8227	8227	8227/M	8227	8227	8227/M	8227	8227
		8227	8227						
local_set	8381	8499/M	8499/M	8499	8499	8499	8499/M	8499/M	8499
		8499	8499/M	8499	8499	8499/M	8499	8499	8499
		8499	8499/M	8499/M	8499	8499	8499	8499/M	8499/M
		8499	8499	8499/M	8499	8499	8499/M	8499	8499
		8499	8499						
local_set	8581	8630/M	8630/M	8630	8630	8630	8630/M	8630/M	8630
		8630	8630/M	8630	8630	8630/M	8630	8630	8630
		8630	8630/M	8630/M	8630	8630	8630	8630/M	8630/M
		8630	8630	8630/M	8630	8630	8630/M	8630	8630
		8630	8630						
local_set	8682	8739/M	8739/M	8739	8739	8739	8739/M	8739/M	8739
		8739	8739/M	8739	8739	8739/M	8739	8739	8739
		8739	8739/M	8739/M	8739	8739	8739	8739/M	8739/M
		8739	8739	8739/M	8739	8739	8739/M	8739	8739
		8739	8739	8739/M	8739/M	8739	8739	8739	8739/M
		8739/M	8739	8739	8739/M	8739	8739	8739/M	8739
		8739	8739	8739					
local_set	8682	8739/M	8739/M	8739	8739	8739/M	8739/M	8739	8739

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8739/M	8739	8739	8739/M	8739	8739	8739	8739
		8739/M	8739/M	8739	8739	8739/M	8739/M	8739	8739
		8739/M	8739	8739	8739/M	8739	8739	8739	8739
		8739/M	8739/M	8739	8739	8739/M	8739/M	8739	8739
		8739/M	8739	8739	8739/M	8739	8739	8739	8739
local_set	9349	9406/M	9406/M	9406	9406	9406	9406/M	9406/M	9406
		9406	9406/M	9406	9406	9406/M	9406	9406	9406
		9406	9406/M	9406/M	9406	9406	9406	9406/M	9406/M
		9406	9406	9406/M	9406	9406	9406/M	9406	9406
		9406	9406	9417/M	9417/M	9417	9417	9417	9417/M
		9417/M	9417	9417	9417/M	9417	9417	9417/M	9417
		9417	9417	9417					
local_set	9446	9479/M	9479/M	9479	9479	9479	9479/M	9479/M	9479
		9479	9479/M	9479	9479	9479/M	9479	9479	9479
		9479	9479/M	9479/M	9479	9479	9479	9479/M	9479/M
		9479	9479	9479/M	9479	9479	9479/M	9479	9479
		9479	9479						
local_set	9491	9528/M	9528/M	9528	9528	9528	9528/M	9528/M	9528
		9528	9528/M	9528	9528	9528/M	9528	9528	9528
		9528	9562/M	9562/M	9562	9562	9562	9562/M	9562/M
		9562	9562	9562/M	9562	9562	9562/M	9562	9562
		9562	9562						
local_set	9584	9623/M	9623/M	9623	9623	9623	9623/M	9623/M	9623
		9623	9623/M	9623	9623	9623/M	9623	9623	9623
		9623	9623/M	9623/M	9623	9623	9623	9623/M	9623/M
		9623	9623	9623/M	9623	9623	9623/M	9623	9623
		9623	9623	9647/M	9647/M	9647	9647	9647	9647/M
		9647/M	9647	9647	9647/M	9647	9647	9647/M	9647
		9647	9647	9647	9647/M	9647/M	9647	9647	9647
		9647/M	9647/M	9647	9647	9647/M	9647	9647	9647/M
		9647	9647	9647	9647				
local_set	9661	9694/M	9694/M	9694	9694	9694	9694/M	9694/M	9694
		9694	9694/M	9694	9694	9694/M	9694	9694	9694
		9694	9694/M	9694/M	9694	9694	9694	9694/M	9694/M
		9694	9694	9694/M	9694	9694	9694/M	9694	9694
		9694	9694	9694/M	9694/M	9694	9694	9694	9694/M
		9694/M	9694	9694	9694/M	9694	9694	9694/M	9694
		9694	9694	9694					
local_set	9661	9694/M	9694/M	9694	9694	9694/M	9694/M	9694	9694
		9694/M	9694	9694	9694/M	9694	9694	9694	9694
		9694/M	9694/M	9694	9694	9694/M	9694/M	9694	9694
		9694/M	9694	9694	9694/M	9694	9694	9694	9694
		9694/M	9694/M	9694	9694	9694/M	9694/M	9694	9694
		9694/M	9694	9694	9694/M	9694	9694	9694	9694
local_set	9753	9766/M	9766/M	9766	9766	9766	9766/M	9766/M	9766
		9766	9766/M	9766	9766	9766/M	9766	9766	9766
		9766	9781/M	9781/M	9781	9781	9781	9781/M	9781/M
		9781	9781	9781/M	9781	9781	9781/M	9781	9781
		9781	9781	9806/M	9806/M	9806	9806	9806	9806/M
		9806/M	9806	9806	9806/M	9806	9806	9806/M	9806
		9806	9806	9806					
local_set	9753	9768/M	9768/M	9768	9768	9768/M	9768/M	9768	9768
		9768/M	9768	9768	9768/M	9768	9768	9768	9768

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9782/M	9782/M	9782	9782	9782/M	9782/M	9782	9782
		9782/M	9782	9782	9782/M	9782	9782	9782	9782
		9807/M	9807/M	9807	9807	9807/M	9807/M	9807	9807
		9807/M	9807	9807	9807/M	9807	9807	9807	9807
local_set	9830	9832/M	9834/M	9835	9835	9836/M	9837/M	9837	9837
		9838/M	9838	9838	9839/M	9839	9839	9840	9841
local_set	9855	9857/M	9858/M	9858	9858	9859	9860/M	9861/M	9861
		9861	9862/M	9862	9862	9863/M	9863	9863	9864
		9866							
local_set	10157	10198/M	10198/M	10198	10198	10198	10198/M	10198/M	10198
		10198	10198/M	10198	10198	10198/M	10198	10198	10198
		10198	10201/M	10201/M	10201	10201	10201	10201/M	10201/M
		10201	10201	10201/M	10201	10201	10201/M	10201	10201
		10201	10201	10201/M	10201/M	10201	10201	10201	10201/M
		10201/M	10201	10201	10201/M	10201	10201	10201/M	10201
		10201	10201	10201					
local_set	10408	10466/M	10466/M	10466	10466	10466	10466/M	10466/M	10466
		10466	10466/M	10466	10466	10466/M	10466	10466	10466
		10466	10466/M	10466/M	10466	10466	10466	10466/M	10466/M
		10466	10466	10466/M	10466	10466	10466/M	10466	10466
		10466	10466	10466/M	10466/M	10466	10466	10466	10466/M
		10466/M	10466	10466	10466/M	10466	10466	10466/M	10466
		10466	10466	10466	10519/M	10519/M	10519	10519	10519
		10519/M	10519/M	10519	10519	10519/M	10519	10519	10519/M
		10519	10519	10519	10519	10519/M	10519/M	10519	10519
		10519	10519/M	10519/M	10519	10519	10519/M	10519	10519
		10519/M	10519	10519	10519	10519	10550/M	10550/M	10550
		10550	10550	10550/M	10550/M	10550	10550	10550/M	10550
		10550	10550/M	10550	10550	10550	10550	10550/M	10550/M
		10550	10550	10550	10550/M	10550/M	10550	10550	10550/M
		10550	10550	10550/M	10550	10550	10550	10550	10579/M
		10579/M	10579	10579	10579	10579/M	10579/M	10579	10579
		10579/M	10579	10579	10579/M	10579	10579	10579	10579
		10579/M	10579/M	10579	10579	10579	10579/M	10579/M	10579
		10579	10579/M	10579	10579	10579/M	10579	10579	10579
		10579	10725/M	10725/M	10725	10725	10725	10725/M	10725/M
		10725	10725	10725/M	10725	10725	10725/M	10725	10725
		10725	10725	10774/M	10774/M	10774	10774	10774	10774/M
		10774/M	10774	10774	10774/M	10774	10774	10774/M	10774
		10774	10774	10774	10774/M	10774/M	10774	10774	10774
		10774/M	10774/M	10774	10774	10774/M	10774	10774	10774/M
		10774	10774	10774	10774	10774/M	10774/M	10774	10774
		10774	10774/M	10774/M	10774	10774	10774/M	10774	10774
		10774/M	10774	10774	10774	10774			
local_set	10408	10466/M	10466/M	10466	10466	10466/M	10466/M	10466	10466
		10466/M	10466	10466	10466/M	10466	10466	10466	10466
		10466/M	10466/M	10466	10466	10466/M	10466/M	10466	10466
		10466/M	10466	10466	10466/M	10466	10466	10466	10466
		10466/M	10466/M	10466	10466	10466/M	10466/M	10466	10466
		10466/M	10466	10466	10466/M	10466	10466	10466	10466
		10774/M	10774/M	10774	10774	10774/M	10774/M	10774	10774
		10774/M	10774	10774	10774/M	10774	10774	10774	10774
		10774/M	10774/M	10774	10774	10774/M	10774/M	10774	10774

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10774/M	10774	10774	10774/M	10774	10774	10774	10774
		10774/M	10774/M	10774	10774	10774/M	10774/M	10774	10774
		10774/M	10774	10774	10774/M	10774	10774	10774	10774
local_set	10429	10657/M	10658/M	10658	10658	10659	10660/M	10661/M	10661
		10661	10662/M	10662	10662	10663/M	10663	10663	10664
		10666	10710/M	10710	10712/M	10712	10714/M	10714	10717
local_set	10879	10925/M	10925/M	10925	10925	10925	10925/M	10925/M	10925
		10925	10925/M	10925	10925	10925/M	10925	10925	10925
		10925	10925/M	10925/M	10925	10925	10925	10925/M	10925/M
		10925	10925	10925/M	10925	10925	10925/M	10925	10925
		10925	10925						
local_set	11095	11139/M	11139/M	11139	11139	11139	11139/M	11139/M	11139
		11139	11139/M	11139	11139	11139/M	11139	11139	11139
		11139	11139/M	11139/M	11139	11139	11139	11139/M	11139/M
		11139	11139	11139/M	11139	11139	11139/M	11139	11139
		11139	11139	11151/M	11151/M	11151	11151	11151	11151/M
		11151/M	11151	11151	11151/M	11151	11151	11151/M	11151
		11151	11151	11151	11151/M	11151/M	11151	11151	11151
		11151/M	11151/M	11151	11151	11151/M	11151	11151	11151/M
		11151	11151	11151	11151				
lock	5286	5295							
lock	5327	5340							
lock	6421	6429	6431	6433/M	6435/M	6435			
lock	6444	6447	6450	6451/M	6451	6453/M			
lock	6464	6490	6490	6490/M	6490/M	6490			
lock	6464	6494	6494	6494/M	6494	6494/M	6502	6502	6502/M
		6502	6502/M						
lock	7416	7450	7450	7450/M	7450/M	7450			
lock	7416	7454	7454	7454/M	7454	7454/M	7466	7466	7466/M
		7466	7466/M						
lock	7472	7501	7501	7501/M	7501/M	7501			
lock	7472	7512	7512	7512/M	7512	7512/M			
lock	7517	7529	7529	7529/M	7529/M	7529			
lock	7517	7547	7547	7547/M	7547	7547/M			
lock	7562	7573	7573	7573/M	7573/M	7573			
lock	7562	7599	7599	7599/M	7599	7599/M			
lock	7618	7653	7653	7653/M	7653/M	7653			
lock	7618	7725	7725	7725/M	7725	7725/M			
lock	7730	7748	7748	7748/M	7748/M	7748			
lock	7730	7757	7757	7757/M	7757	7757/M			
lock	7766	7806	7806	7806/M	7806/M	7806			
lock	7766	7827	7827	7827/M	7827	7827/M			
lock	7932	7956	7956	7956/M	7956/M	7956			
lock	7932	7983	7983	7983/M	7983	7983/M			
lock	8011	8017	8017	8017/M	8017/M	8017			
lock	8011	8039	8039	8039/M	8039	8039/M			
lock	8044	8070	8070	8070/M	8070/M	8070			
lock	8044	8104	8104	8104/M	8104	8104/M			
lock	8110	8134	8134	8134/M	8134/M	8134			
lock	8110	8176	8176	8176/M	8176	8176/M			
lock	8182	8222	8232						
lock	8259	8265	8265	8265/M	8265/M	8265			
lock	8259	8270	8270	8270/M	8270	8270/M			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	8276	8282	8282	8282/M	8282/M	8282			
lock	8276	8292	8292	8292/M	8292	8292/M			
lock	8381	8444	8444	8444/M	8444/M	8444			
lock	8381	8451	8451	8451/M	8451	8451/M	8508	8508	8508/M
		8508	8508/M						
lock	8513	8532	8532	8532/M	8532/M	8532			
lock	8513	8553	8553	8553/M	8553	8553/M			
lock	8581	8607	8607	8607/M	8607/M	8607			
lock	8581	8610	8610	8610/M	8610	8610/M	8677	8677	8677/M
		8677	8677/M						
lock	8682	8727	8727	8727/M	8727/M	8727			
lock	8682	8756	8756	8756/M	8756	8756/M			
lock	8770	8783	8783	8783/M	8783/M	8783			
lock	8770	8870	8870	8870/M	8870	8870/M			
lock	8875	8897	8897	8897/M	8897/M	8897			
lock	8875	8961	8961	8961/M	8961	8961/M			
lock	8966	9004	9004	9004/M	9004/M	9004			
lock	8966	9010	9010	9010/M	9010	9010/M	9028	9028	9028/M
		9028	9028/M	9048	9048	9048/M	9048	9048/M	
lock	9057	9100	9100	9100/M	9100/M	9100			
lock	9057	9104	9104	9104/M	9104	9104/M	9211	9211	9211/M
		9211	9211/M						
lock	9217	9247	9247	9247/M	9247/M	9247			
lock	9217	9250	9250	9250/M	9250	9250/M	9257	9257	9257/M
		9257	9257/M	9271	9271	9271/M	9271	9271/M	
lock	9322	9329	9329	9329/M	9329/M	9329			
lock	9322	9339	9339	9339/M	9339	9339/M			
lock	9446	9469	9469	9469/M	9469/M	9469			
lock	9446	9486	9486	9486/M	9486	9486/M			
lock	9491	9504	9504	9504/M	9504/M	9504			
lock	9491	9539	9539	9539/M	9539	9539/M	9573	9573	9573/M
		9573	9573/M						
lock	9584	9607	9607	9607/M	9607/M	9607			
lock	9584	9631	9631	9631/M	9631	9631/M	9655	9655	9655/M
		9655	9655/M						
lock	9936	9951	9951	9951/M	9951/M	9951			
lock	9936	9975	9975	9975/M	9975	9975/M	10137	10137	10137/M
		10137	10137/M	10141	10141	10141/M	10141	10141/M	
lock	10157	10170	10170	10170/M	10170/M	10170			
lock	10157	10210	10210	10210/M	10210	10210/M			
lock	10224	10233	10233	10233/M	10233/M	10233			
lock	10224	10246	10246	10246/M	10246	10246/M			
lock	10408	10461	10461	10461/M	10461/M	10461	10593	10593	10593/M
		10593/M	10593						
lock	10408	10523	10523	10523/M	10523	10523/M	10589	10589	10589/M
		10589	10589/M	10820	10820	10820/M	10820	10820/M	
lock	10964	10981	10981	10981/M	10981/M	10981			
lock	10964	10988	10988	10988/M	10988	10988/M	11022	11022	11022/M
		11022	11022/M						
lock	11095	11107	11107	11107/M	11107/M	11107			
lock	11095	11155	11155	11155/M	11155	11155/M			
locked	4034	5295	5340	8222	8232				
locked	4737	5671	6431	6490	7450	7501	7529	7573	7653

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7748	7806	7956	8017	8070	8134	8265	8282
		8444	8532	8607	8727	8783	8897	9004	9100
		9247	9329	9469	9504	9607	9951	10170	10233
		10461	10593	10981	11107				
locked_dp_integer	10424	10484/M	10489/M	10521					
long_wait_aging_complete	1277	7926/M	9140	9145/M	9237	9239/M	10550/M	10914/M	
low_prio_csti	6877	6943/M	6952/M	6959	6960/S	6965/S	6972/S	6973/S	6974/S
		6975/S	6982/S	6987/S	6988/S	6990/S	6993	6994/S	6995/S
low_prio_csti	7766	7866/M	7866/M	7866	7866/S	7866/S	7866/S	7866/S	7866/S
		7866/S	7866/S	7866/S	7866/S	7866/S	7866	7866/S	7866/S
		7898/M	7898/M	7898	7898/S	7898/S	7898/S	7898/S	7898/S
		7898/S	7898/S	7898/S	7898/S	7898/S	7898	7898/S	7898/S
low_prio_csti	7932	7979/M	7979/M	7979	7979/S	7979/S	7979/S	7979/S	7979/S
		7979/S	7979/S	7979/S	7979/S	7979/S	7979	7979/S	7979/S
low_prio_csti	8110	8172/M	8172/M	8172	8172/S	8172/S	8172/S	8172/S	8172/S
		8172/S	8172/S	8172/S	8172/S	8172/S	8172	8172/S	8172/S
low_prio_csti	8182	8227/M	8227/M	8227	8227/S	8227/S	8227/S	8227/S	8227/S
		8227/S	8227/S	8227/S	8227/S	8227/S	8227	8227/S	8227/S
low_prio_csti	8381	8499/M	8499/M	8499	8499/S	8499/S	8499/S	8499/S	8499/S
		8499/S	8499/S	8499/S	8499/S	8499/S	8499	8499/S	8499/S
low_prio_csti	8581	8630/M	8630/M	8630	8630/S	8630/S	8630/S	8630/S	8630/S
		8630/S	8630/S	8630/S	8630/S	8630/S	8630	8630/S	8630/S
low_prio_csti	9349	9406/M	9406/M	9406	9406/S	9406/S	9406/S	9406/S	9406/S
		9406/S	9406/S	9406/S	9406/S	9406/S	9406	9406/S	9406/S
low_prio_csti	9446	9479/M	9479/M	9479	9479/S	9479/S	9479/S	9479/S	9479/S
		9479/S	9479/S	9479/S	9479/S	9479/S	9479	9479/S	9479/S
low_prio_csti	9584	9623/M	9623/M	9623	9623/S	9623/S	9623/S	9623/S	9623/S
		9623/S	9623/S	9623/S	9623/S	9623/S	9623	9623/S	9623/S
		9647/M	9647/M	9647	9647/S	9647/S	9647/S	9647/S	9647/S
		9647/S	9647/S	9647/S	9647/S	9647/S	9647	9647/S	9647/S
low_prio_csti	10157	10201/M	10201/M	10201	10201/S	10201/S	10201/S	10201/S	10201/S
		10201/S	10201/S	10201/S	10201/S	10201/S	10201	10201/S	10201/S
low_prio_csti	10408	10519/M	10519/M	10519	10519/S	10519/S	10519/S	10519/S	10519/S
		10519/S	10519/S	10519/S	10519/S	10519/S	10519	10519/S	10519/S
		10550/M	10550/M	10550	10550/S	10550/S	10550/S	10550/S	10550/S
		10550/S	10550/S	10550/S	10550/S	10550/S	10550	10550/S	10550/S
		10579/M	10579/M	10579	10579/S	10579/S	10579/S	10579/S	10579/S
		10579/S	10579/S	10579/S	10579/S	10579/S	10579	10579/S	10579/S
low_prio_csti	10879	10925/M	10925/M	10925	10925/S	10925/S	10925/S	10925/S	10925/S
		10925/S	10925/S	10925/S	10925/S	10925/S	10925	10925/S	10925/S
low_prio_csti	11095	11139/M	11139/M	11139	11139/S	11139/S	11139/S	11139/S	11139/S
		11139/S	11139/S	11139/S	11139/S	11139/S	11139	11139/S	11139/S
		11151/M	11151/M	11151	11151/S	11151/S	11151/S	11151/S	11151/S
		11151/S	11151/S	11151/S	11151/S	11151/S	11151	11151/S	11151/S
low_priority	6876	6940/M	6947	6953/M	6956	6993			
low_priority	7766	7866/M	7866	7866/M	7866	7866	7898/M	7898	7898/M
		7898	7898						
low_priority	7932	7979/M	7979	7979/M	7979	7979			
low_priority	8110	8172/M	8172	8172/M	8172	8172			
low_priority	8182	8227/M	8227	8227/M	8227	8227			
low_priority	8381	8499/M	8499	8499/M	8499	8499			
low_priority	8581	8630/M	8630	8630/M	8630	8630			
low_priority	9349	9406/M	9406	9406/M	9406	9406			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
low_priority	9446	9479/M	9479	9479/M	9479	9479			
low_priority	9584	9623/M	9623	9623/M	9623	9623	9647/M	9647	9647/M
		9647	9647						
low_priority	10157	10201/M	10201	10201/M	10201	10201			
low_priority	10408	10519/M	10519	10519/M	10519	10519	10550/M	10550	10550/M
		10550	10550	10579/M	10579	10579/M	10579	10579	
low_priority	10879	10925/M	10925	10925/M	10925	10925			
low_priority	11095	11139/M	11139	11139/M	11139	11139	11151/M	11151	11151/M
		11151	11151						
major	1483	6895	6900/M	6900	7866	7866/M	7866	7898	7898/M
		7898	7979	7979/M	7979	8172	8172/M	8172	8227
		8227/M	8227	8499	8499/M	8499	8630	8630/M	8630
		8739	8739/M	8739/M	9406	9406/M	9406	9479	9479/M
		9479	9623	9623/M	9623	9647	9647/M	9647	9726
		9733/M	9735/M	10201	10201/M	10201	10466	10466/M	10466/M
		10519	10519/M	10519	10550/M	10550	10550/M	10550	10579
		10579/M	10579	10923/M	10925	10925/M	10925	11139	11139/M
		11139	11151	11151/M	11151				
major_priority	5102	6691/M	6697	6898/M	6716	6717/M	6975/M	7011/M	7026
		7027/M	7034/S	7035/S	7036/M	7054	7054/M	7054	7054/M
		7056/M	7056	7056/M	7058	7058/M	7130	7131/M	7165
		7166/M	7242/M	7274	7275/M	7329	7330/M	7864/M	7864
		7864/M	7864	7864/M	7866/M	7866/M	7866	7866/M	7866/S
		7866/S	7866/M	7866	7866/M	7866	7866/M	7866/M	7866
		7866/M	7866	7866/M	7898/M	7898/M	7898	7898/M	7898/S
		7898/S	7898/M	7898	7898/M	7898	7898/M	7898/M	7898
		7898/M	7898	7898/M	7979/M	7979/M	7979	7979/M	7979/S
		7979/S	7979/M	7979	7979/M	7979	7979/M	7979/M	7979
		7979/M	7979	7979/M	8172/M	8172/M	8172	8172/M	8172/S
		8172/S	8172/M	8172	8172/M	8172	8172/M	8172/M	8172
		8172/M	8172	8172/M	8227/M	8227/M	8227	8227/M	8227/S
		8227/S	8227/M	8227	8227/M	8227	8227/M	8227/M	8227
		8227/M	8227	8227/M	8499/M	8499/M	8499	8499/M	8499/S
		8499/S	8499/M	8499	8499/M	8499	8499/M	8499/M	8499
		8499/M	8499	8499/M	8630/M	8630/M	8630	8630/M	8630/S
		8630/S	8630/M	8630	8630/M	8630	8630/M	8630/M	8630
		8630/M	8630	8630/M	9406/M	9406/M	9406	9406/M	9406/S
		9406/S	9406/M	9406	9406/M	9406	9406/M	9406/M	9406
		9406/M	9406	9406/M	9417/M	9417	9417/M	9417	9417/M
		9479/M	9479/M	9479	9479/M	9479/S	9479/S	9479/M	9479
		9479/M	9479	9479/M	9479/M	9479	9479/M	9479	9479/M
		9528/M	9528	9528/M	9528	9528/M	9562/M	9562	9562/M
		9562	9562/M	9623/M	9623/M	9623	9623/M	9623/S	9623/S
		9623/M	9623	9623/M	9623	9623/M	9623/M	9623	9623/M
		9623	9623/M	9647/M	9647/M	9647	9647/M	9647/S	9647/S
		9647/M	9647	9647/M	9647	9647/M	9647/M	9647	9647/M
		9647	9647/M	10198/M	10198	10198/M	10198	10198/M	10201/M
		10201/M	10201	10201/M	10201/S	10201/S	10201/M	10201	10201/M
		10201	10201/M	10201/M	10201	10201/M	10201	10201/M	10519/M
		10519/M	10519	10519/M	10519/S	10519/S	10519/M	10519	10519/M
		10519	10519/M	10519/M	10519	10519/M	10519	10519/M	10550/M
		10550/M	10550	10550/M	10550/S	10550/S	10550/M	10550	10550/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10550	10550/M	10550/M	10550	10550/M	10550	10550/M	10579/M
		10579/M	10579	10579/M	10579/S	10579/S	10579/M	10579	10579/M
		10579	10579/M	10579/M	10579	10579/M	10579	10579/M	10725/M
		10725	10725/M	10725	10725/M	10925/M	10925/M	10925	10925/M
		10925/S	10925/S	10925/M	10925	10925/M	10925	10925/M	10925/M
		10925	10925/M	10925	10925/M	11139/M	11139/M	11139	11139/M
		11139/S	11139/S	11139/M	11139	11139/M	11139	11139/M	11139/M
		11139	11139/M	11139	11139/M	11151/M	11151/M	11151	11151/M
		11151/S	11151/S	11151/M	11151	11151/M	11151	11151/M	11151/M
		11151	11151/M	11151	11151/M				
major_time_slice_remaining	8682	8739/M	8739	8739					
major_time_slice_remaining	9677	9726/M	9732	9735					
major_time_slice_remaining	10408	10466/M	10466	10466					
major_timeslice_insert	6873	6896	7033	7055					
major_timeslice_insert	7766	7866	7866	7866	7898	7898	7898		
major_timeslice_insert	7932	7979	7979	7979					
major_timeslice_insert	8110	8172	8172	8172					
major_timeslice_insert	8182	8227	8227	8227					
major_timeslice_insert	8381	8499	8499	8499					
major_timeslice_insert	8581	8630	8630	8630					
major_timeslice_insert	9349	9406	9406	9406					
major_timeslice_insert	9446	9479	9479	9479					
major_timeslice_insert	9584	9623	9623	9623	9647	9647	9647		
major_timeslice_insert	10157	10201	10201	10201					
major_timeslice_insert	10408	10519	10519	10519	10550	10550	10550	10579	10579
		10579							
major_timeslice_insert	10879	10925	10925	10925					
major_timeslice_insert	11095	11139	11139	11139	11151	11151	11151		
max_block_in_use	4064	7691							
max_cptime	1160	9000/M	10444/M	10496/M	10759/M	10786/M			
maximum_time	4804	8739	8739/M	8739	9694	9694/M	9694	9794	9809/M
		9811	10466	10466/M	10466	10774	10774/M	10774	
maximums_defined	4788	8739	9694	9790	10466	10774			
maximums_exceeded	4818	6694	6694/M	6694	6694	6979	6979/M	6979	6979
		7014	7014/M	7014	7014	7864	7864/M	7864	7864
		7866	7866/M	7866	7866	7866	7866/M	7866	7866
		7898	7898/M	7898	7898	7898	7898/M	7898	7898
		7979	7979/M	7979	7979	7979	7979/M	7979	7979
		8172	8172/M	8172	8172	8172	8172/M	8172	8172
		8227	8227/M	8227	8227	8227	8227/M	8227	8227
		8499	8499/M	8499	8499	8499	8499/M	8499	8499
		8630	8630/M	8630	8630	8630	8630/M	8630	8630
		8739/M	8739	8739/M	8739	8739	8739	8739/M	8739
		8739	8739	8739/M	8739	8739	8739	8739/M	8739
		8739	8739	8739/M	8739	8739	8739/M	8739	8739
		8739	8739/M	8739	8739	9406	9406/M	9406	9406
		9406	9406/M	9406	9406	9417	9417/M	9417	9417
		9479	9479/M	9479	9479	9479	9479/M	9479	9479
		9528	9528/M	9528	9528	9562	9562/M	9562	9562
		9623	9623/M	9623	9623	9623	9623/M	9623	9623
		9647	9647/M	9647	9647	9647	9647/M	9647	9647
		9694/M	9694	9694/M	9694	9694	9694	9694/M	9694
		9694	9694	9694/M	9694	9694	9694	9694/M	9694

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9694	9694	9694/M	9694	9694	9694/M	9694	9694
		9694	9694/M	9694	9694	9764/M	9766	9766/M	9766
		9766	9768	9768/M	9768	9768	9781	9781/M	9781
		9781	9782	9782/M	9782	9782	9792	9795/M	9796
		9806	9806/M	9806	9806	9807	9807/M	9807	9807
		9835	9838/M	9838	9840	9859	9862/M	9862	9864
		10198	10198/M	10198	10198	10201	10201/M	10201	10201
		10201	10201/M	10201	10201	10466/M	10466	10466/M	10466
		10466	10466	10466/M	10466	10466	10466	10466/M	10466
		10466	10466	10466/M	10466	10466	10466	10466/M	10466
		10466	10466/M	10466	10466	10466	10466/M	10466	10466
		10519	10519/M	10519	10519	10519	10519/M	10519	10519
		10550	10550/M	10550	10550	10550	10550/M	10550	10550
		10579	10579/M	10579	10579	10579	10579/M	10579	10579
		10659	10662/M	10662	10664	10714/M	10714	10725	10725/M
		10725	10725	10774/M	10774	10774/M	10774	10774	10774
		10774/M	10774	10774	10774	10774/M	10774	10774	10774
		10774/M	10774	10774	10774	10774/M	10774	10774	10774/M
		10774	10774	10774	10774/M	10774	10774	10925	10925/M
		10925	10925	10925	10925/M	10925	10925	11139	11139/M
		11139	11139	11139	11139/M	11139	11139	11151	11151/M
		11151	11151	11151	11151/M	11151	11151		
maximums_exceeded	6180	8739/M	8739	9694/M	9694	9798/M	9799	10466/M	10466
		10774/M	10774						
maxws_aio_slowdown_display	1305	8728/M							
memory_port_mask	1147	6995/P	7866/P	7898/P	7979/P	8172/P	8227/P	8499/P	8630/P
		9406/P	9479/P	9623/P	9647/P	10201/P	10519/P	10550/P	10579/P
		10925/P	11139/P	11151/P					
minimum	2591	9201	9205						
minimum_time	4803	8739	8739/M	8739	9694	9694/M	9694	9777	9784/M
		9786	10466	10466/M	10466	10774	10774/M	10774	
minimums_to_satisfy	4787	8739	9694	9763	10466	10774			
minimums_to_satisfy	4816	6694/M	6694	6694	6979/M	6979	6979	7014/M	7014
		7014	7864/M	7864	7864	7866/M	7866	7866	7866/M
		7866	7866	7898/M	7898	7898	7898/M	7898	7898
		7979/M	7979	7979	7979/M	7979	7979	8172/M	8172
		8172	8172/M	8172	8172	8227/M	8227	8227	8227/M
		8227	8227	8499/M	8499	8499	8499/M	8499	8499
		8630/M	8630	8630	8630/M	8630	8630	8739/M	8739/M
		8739	8739	8739/M	8739	8739	8739	8739/M	8739
		8739/M	8739	8739	8739/M	8739	8739	8739/M	8739
		8739	8739/M	8739	8739	9406/M	9406	9406	9406/M
		9406	9406	9417/M	9417	9417	9479/M	9479	9479
		9479/M	9479	9479	9528/M	9528	9528	9562/M	9562
		9562	9623/M	9623	9623	9623/M	9623	9623	9647/M
		9647	9647	9647/M	9647	9647	9694/M	9694/M	9694
		9694	9694/M	9694	9694	9694	9694/M	9694	9694/M
		9694	9694	9694/M	9694	9694	9694/M	9694	9694
		9694/M	9694	9694	9763/M	9766/M	9766	9766	9768/M
		9768	9768	9775	9778/M	9779	9781/M	9781	9781
		9782/M	9782	9782	9806/M	9806	9806	9807/M	9807
		9807	9837/M	9837	9839	9861/M	9861	9863	10198/M
		10198	10198	10201/M	10201	10201	10201/M	10201	10201

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10466/M	10466/M	10466	10466	10466/M	10466	10466	10466
		10466/M	10466	10466/M	10466	10466	10466/M	10466	10466
		10466/M	10466	10466	10466/M	10466	10466	10519/M	10519
		10519	10519/M	10519	10519	10550/M	10550	10550	10550/M
		10550	10550	10579/M	10579	10579	10579/M	10579	10579
		10661/M	10661	10663	10710/M	10710	10725/M	10725	10725
		10774/M	10774/M	10774	10774	10774/M	10774	10774	10774
		10774/M	10774	10774/M	10774	10774	10774/M	10774	10774
		10774/M	10774	10774	10774/M	10774	10774	10925/M	10925
		10925	10925/M	10925	10925	11139/M	11139	11139	11139/M
		11139	11139	11151/M	11151	11151	11151/M	11151	11151
minor	1482	6893	6893	6895	6897/M	6897	6901/M	6901	7866
		7866	7866	7866/M	7866	7866/M	7866	7898	7898
		7898	7898/M	7898	7898/M	7898	7979	7979	7979
		7979/M	7979	7979/M	7979	8172	8172	8172	8172/M
		8172	8172/M	8172	8227	8227	8227	8227/M	8227
		8227/M	8227	8499	8499	8499	8499/M	8499	8499/M
		8499	8630	8630	8630	8630/M	8630	8630/M	8630
		8739	8739/M	8739/M	9406	9406	9406	9406/M	9406
		9406/M	9406	9479	9479	9479	9479/M	9479	9479/M
		9479	9623	9623	9623	9623/M	9623	9623/M	9623
		9647	9647	9647	9647/M	9647	9647/M	9647	9725
		9728/M	9730/M	10201	10201	10201	10201/M	10201	10201/M
		10201	10466	10466/M	10466/M	10498	10519	10519	10519
		10519/M	10519	10519/M	10519	10550/M	10550	10550	10550
		10550/M	10550	10550/M	10550	10579	10579	10579	10579/M
		10579	10579/M	10579	10786	10922/M	10925	10925	10925
		10925/M	10925	10925/M	10925	11139	11139	11139	11139/M
		11139	11139/M	11139	11151	11151	11151	11151/M	11151
		11151/M	11151						
minor_priority	5101	6690/M	6700	6701/M	6719	6720/M	6974/M	7010/M	7020/S
		7021/S	7026	7029/M	7054	7054/M	7054	7054/M	7056
		7056/M	7056	7056/M	7058	7058/M	7130	7136/M	7162
		7163/M	7226	7227/M	7271	7272/M	7326	7327/M	7864/M
		7864	7864/M	7864	7864/M	7866/M	7866/M	7866/S	7866/S
		7866	7866/M	7866	7866/M	7866	7866/M	7866	7866/M
		7866	7866/M	7866	7866/M	7898/M	7898/M	7898/S	7898/S
		7898	7898/M	7898	7898/M	7898	7898/M	7898	7898/M
		7898	7898/M	7898	7898/M	7979/M	7979/M	7979/S	7979/S
		7979	7979/M	7979	7979/M	7979	7979/M	7979	7979/M
		7979	7979/M	7979	7979/M	8172/M	8172/M	8172/S	8172/S
		8172	8172/M	8172	8172/M	8172	8172/M	8172	8172/M
		8172	8172/M	8172	8172/M	8227/M	8227/M	8227/S	8227/S
		8227	8227/M	8227	8227/M	8227	8227/M	8227	8227/M
		8227	8227/M	8227	8227/M	8499/M	8499/M	8499/S	8499/S
		8499	8499/M	8499	8499/M	8499	8499/M	8499	8499/M
		8499	8499/M	8499	8499/M	8630/M	8630/M	8630/S	8630/S
		8630	8630/M	8630	8630/M	8630	8630/M	8630	8630/M
		8630	8630/M	8630	8630/M	9406/M	9406/M	9406/S	9406/S
		9406	9406/M	9406	9406/M	9406	9406/M	9406	9406/M
		9406	9406/M	9406	9406/M	9417/M	9417	9417/M	9417
		9417/M	9479/M	9479/M	9479/S	9479/S	9479	9479/M	9479
		9479/M	9479	9479/M	9479	9479/M	9479	9479/M	9479

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9479/M	9528/M	9528	9528/M	9528	9528/M	9562/M	9562
		9562/M	9562	9562/M	9623/M	9623/M	9623/S	9623/S	9623
		9623/M	9623	9623/M	9623	9623/M	9623	9623/M	9623
		9623/M	9623	9623/M	9647/M	9647/M	9647/S	9647/S	9647
		9647/M	9647	9647/M	9647	9647/M	9647	9647/M	9647
		9647/M	9647	9647/M	10198/M	10198	10198/M	10198	10198/M
		10201/M	10201/M	10201/S	10201/S	10201	10201/M	10201	10201/M
		10201	10201/M	10201	10201/M	10201	10201/M	10201	10201/M
		10519/M	10519/M	10519/S	10519/S	10519	10519/M	10519	10519/M
		10519	10519/M	10519	10519/M	10519	10519/M	10519	10519/M
		10550/M	10550/M	10550/S	10550/S	10550	10550/M	10550	10550/M
		10550	10550/M	10550	10550/M	10550	10550/M	10550	10550/M
		10579/M	10579/M	10579/S	10579/S	10579	10579/M	10579	10579/M
		10579	10579/M	10579	10579/M	10579	10579/M	10579	10579/M
		10725/M	10725	10725/M	10725	10725/M	10925/M	10925/M	10925/S
		10925/S	10925	10925/M	10925	10925/M	10925	10925/M	10925
		10925/M	10925	10925/M	10925	10925/M	11139/M	11139/M	11139/S
		11139/S	11139	11139/M	11139	11139/M	11139	11139/M	11139
		11139/M	11139	11139/M	11139	11139/M	11151/M	11151/M	11151/S
		11151/S	11151	11151/M	11151	11151/M	11151	11151/M	11151
		11151/M	11151	11151/M	11151	11151/M			
minor_time_slice_remaining	8682	8739/M	8739	8739					
minor_time_slice_remaining	9676	9725/M	9727	9730					
minor_time_slice_remaining	10408	10466/M	10466	10466					
minor_timeslice_insert	6873	6894	6922	7019	7053				
minor_timeslice_insert	7766	7866	7866	7866	7866	7898	7898	7898	7898
minor_timeslice_insert	7932	7979	7979	7979	7979				
minor_timeslice_insert	8110	8172	8172	8172	8172				
minor_timeslice_insert	8182	8227	8227	8227	8227				
minor_timeslice_insert	8381	8499	8499	8499	8499				
minor_timeslice_insert	8581	8630	8630	8630	8630				
minor_timeslice_insert	9349	9406	9406	9406	9406				
minor_timeslice_insert	9446	9479	9479	9479	9479				
minor_timeslice_insert	9584	9623	9623	9623	9623	9647	9647	9647	9647
minor_timeslice_insert	10157	10201	10201	10201	10201				
minor_timeslice_insert	10408	10519	10519	10519	10519	10550	10550	10550	10550
		10579	10579	10579	10579				
minor_timeslice_insert	10879	10925	10925	10925	10925				
minor_timeslice_insert	11095	11139	11139	11139	11139	11151	11151	11151	11151
mlc$error	298	300	307	315	326	334	342	349	357
		364	373	381	390	406	414	420	429
		437	445	454	462	469	479	489	497
		507	514	522	531	535			
mlc$fetch_link_partner_info_req	601	605							
mlc$max_message_length	286	84	559						
mlc$max_queued_messages	292	558	560	561	562				
mlc$nosve_not_up	406	589							
mlc$ok	414	589							
mlc$sign_on_req	593	605							
mlc$unique_name	289	290							
mlimi	70	74	84						
mlt$application_name	556	582	585						
mlt$c170_rqst_blk	73	6348							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mlt$direction	575	579							
mlt$receive_entry	564	563							
mlt$signal	573	587							
mlt$signal_record	576	573							
mlt$signaler_application_info	581	586							
mlv$c170_rqst_blk	6348	7902	7903	9005	9006	10452/P			
mmc$	4334	4340	4343	4346	4349	4352	4355	4358	4362
		4365	4368	4371	4374	4377	4380	4384	4387
		4390	4393	4396	4399	4403	4406	4409	4412
		4415	4418	4421	4424	4427	4430	4433	4436
		4439	4442	4445	4448	4451	4454	4457	4460
		4463	4467	4470	4473	4476	4479	4482	4485
		4488	4491	4494	4497	4500	4503	4506	4509
		4512	4515	4518	4521	4524	4527	4531	4535
		4538	4542	4545	4548	4551	4554	4557	4560
		4563	4566	4569	4572	4575	4578	4581	4584
		4587	4590	4593	4596	4599	4603	4607	4610
		4613	4616	4619	4622	4625	4628	4631	4634
		4637	4640	4643	4646	4649	4652	4655	4658
		4661	4665	4668	4671				
mmc$assign_active_null	2775	2776							
mmc$cell_pointer	2874	2879							
mmc$heap_pointer	2875	2883							
mmc$kw_asid	2800	2836							
mmc$kw_clear_space	2798	2823							
mmc$kw_current_segment_length	2797	2817							
mmc$kw_error_exit_procedure	2799	2827							
mmc$kw_gl_key	2799	2821							
mmc$kw_hardware_attributes	2801	2830							
mmc$kw_inheritance	2801	2838							
mmc$kw_max_segment_length	2798	2819							
mmc$kw_preset_value	2800	2825							
mmc$kw_ps_transfer_size	2802	2846							
mmc$kw_ring_numbers	2796	2812							
mmc$kw_segment_access_control	2800	2834							
mmc$kw_segment_number	2797	2815							
mmc$kw_shadow_segment	2802	2840							
mmc$kw_software_attributes	2799	2832							
mmc$kw_wired_segment	2802	2843							
mmc$lus_lock_segment	4716	4717							
mmc$lus_unlock_segment	4716	4719							
mmc$mf_segment_mgr_flag	3364	8855/P							
mmc$pq_avail	1830	1876							
mmc$pq_free	1829	1888							
mmc$pq_job_fixed	1870	1877	1889						
mmc$pq_job_working_set	1872	1889	1890						
mmc$pq_shared_first_site	1880	1884							
mmc$pq_shared_num_sites	1881	1884							
mmc$pq_shared_other	1839	1879							
mmc$pq_shared_site_01	1841	1880							
mmc$pq_shared_site_25	1885	1885							
mmc$pq_shared_task_service	1834	1878							
mmc$pq_swapped_io_error	1868	1888							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$pq_wired	1832	1875							
mmc$segment_fault_processor_id	3383	3437							
mmc$sequence_pointer	2874	2881							
mmc$ssk_none	2975	2947							
mmc$ssk_segment_number	2976	2945							
mmp$create_job	5552	8493							
mmp$create_task	5540	8675							
mmp$exit_job	5561	8567							
mmp$exit_task	5547	8723							
mmp$get_sdtx_entry_p	5566	8221							
mmp$get_sdtx_entry_p	5568	5570/M	8221/M						
mmt$active_segment_table	2106	6321							
mmt$active_segment_table_entry	2091	2069	2107	2140					
mmt$ast_index	2123	1337	2084	2669	3886	8206			
mmt$attribute_keyword	2796	2811							
mmt$eoi_state	4022	3888							
mmt$global_page_queue_index	1888	2213							
mmt$global_page_queue_list_ent	2203	2213							
mmt$hardware_attribute_set	2865	2831							
mmt$hardware_attributes	2853	2865							
mmt$job_page_queue_index	1889	1815	2214						
mmt$job_page_queue_list	2214	1292							
mmt$link	2114	2092	2130	2131	2200				
mmt$lock_segment_status	2955	2740							
mmt$locked_page	2152	2136							
mmt$lus_lock_type	2905	4718							
mmt$lus_page_disposition	2907	4720							
mmt$max_sdt	2679	2683							
mmt$max_sdt_p	2683	8401							
mmt$max_sdtx	2764	2768							
mmt$memory_reserve_request	2181	1285							
mmt$page_age	2159	2139	2163	2163					
mmt$page_frame_index	2052	2044	2046	2047	2048	2116	2116	2183	2184
		5968	5970	5972					
mmt$page_frame_queue_id	1890	2045	2100	2134					
mmt$page_frame_table_entry	2129	2067	2145						
mmt$page_queue_list_entry	2199	2204	2214						
mmt$rb_lock_unlock_segment	4710	8213							
mmt$sdtx_stream_data	2747	2743							
mmt$segment_access_condition	3410	3438							
mmt$segment_access_rights	2919	2739							
mmt$segment_access_state	2925	2734							
mmt$segment_descriptor	2666	2676	2680	5924					
mmt$segment_descriptor_extended	2732	2761	2765	5568	5571	8211	8221		
mmt$segment_inheritance	2782	2736	2839						
mmt$segment_pointer_kind	2874	2878							
mmt$segment_reservation_state	2965	2737							
mmt$shadow_info	2940	2741							
mmt$shadow_reference_info	2988	2556							
mmt$shadow_segment_kind	2975	2944							
mmt$software_attribute_set	2867	2738	2833						
mmt$software_attributes	2861	2867							
mmt$xcb_page_wait_info	2999	2542							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmv$ast_p	6321	8230	8230	8231	8232/P				
monitor_cptime	4869	9921/M							
monitor_flags	2510	9023/M	9023	9157/M	9157	10129/M	10129	10793/M	10793
		10847/M	10847						
monitor_flags	4939	6614/M	6616/M	8445/M	8445/M	8608/M	8608/M	9020	9023
		9026/M	9141	9154	9157	9159/M	10130	10134/M	10790
		10793	10794/M	10870/M	10870	11004/M			
monitor_lock	3876	5295/P	8222/P	8232/P					
mtc$job_fixed_segment	4122	5252	5293	7054	7054	7056	7056	7058	7106
		7149	7218	7258	7320	7463	7509	7543	7581
		7612	7721	7810	7866	7866	7866	7866	7866
		7898	7898	7898	7898	7898	7978	7979	7979
		7979	7979	7979	8170	8172	8172	8172	8172
		8172	8218	8222	8227	8227	8227	8227	8227
		8232	8425	8437	8459	8499	8499	8499	8499
		8499	8606	8630	8630	8630	8630	8630	8754
		9255	9302	9368	9385	9406	9406	9406	9406
		9406	9471	9479	9479	9479	9479	9479	9515
		9549	9612	9623	9623	9623	9623	9623	9636
		9647	9647	9647	9647	9647	9887	10201	10201
		10201	10201	10201	10516	10519	10519	10519	10519
		10519	10530	10550	10550	10550	10550	10550	10550
		10579	10579	10579	10579	10579	10638	10639	10681
		10682	10921	10925	10925	10925	10925	10925	11139
		11139	11139	11139	11139	11150	11151	11151	11151
		11151	11151						
mtc$scb_max_hardware_status	5889	5839							
mtp$cst_p	5633	6941	7585	7747	7820	7866	7898	7979	8004
		8016	8091	8172	8227	8499	8630	9406	9479
		9623	9647	10201	10519	10550	10579	10925	11139
		11151							
mtp$error_stop	5323	6499	6537	6552	6686	6709	6756	6769	7045
		7452	7502	7574	7579	7751	7807	7864	7864
		7866	7898	7957	7965	7979	8020	8072	8076
		8079	8097	8138	8158	8172	8227	8246	8266
		8288	8334	8499	8534	8537	8630	8710	8725
		8789	8793	8796	9102	9406	9417	9417	9479
		9528	9528	9562	9562	9623	9647	10138	10198
		10198	10201	10519	10550	10550	10561	10579	10590
		10725	10725	10904	10925	10997	10997	11079	11124
		11139	11151						
mtp$interrupt_processor	5528	6995	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10519	10550	10579
		10925	11139	11151					
mtp$set_interlock	5327	5295	5348	8222	8232				
mtp$set_status_abnormal	5645	5652	6496	6534	6604	7452	7461	8445	8608
		8704	9102	9373	9412	9433	9543	9577	9976
mtt$idle_status_block	5817	5804							
mtt$monitor_interlock	4029	3876	5327						
mtt$scb_180_status	5793	5784							
mtt$scb_hardware_status	5841	5779	5864						
mtt$scb_hardware_status_count	5839	5842							
mtt$scb_hardware_status_msg	5845	5851							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mtt$scb_hardware_status_msgs	5850	5787						
mtt$scb_hardware_status_options	5831	5841	5850					
mtt$smu_communications_block	5778	5771						
mtt$step_status_block	5812	5805						
mtt$system_idle_update_request	5894	5819	5819					
mtt$system_status_block	5803	5794						
mtt$system_step_update_request	5893	5814	5814					
mtv$cst0	5639	5635	6941	6947	6953	6960	6965/S	6972	6973
		6974	6975	6982	6987/M	6988/M	6990/M	6994	6995/P
		7585	7747	7820	7866	7866	7866	7866	7866/S
		7866	7866	7866	7866	7866	7866/M	7866/M	7866/M
		7866	7866/P	7898	7898	7898	7898	7898/S	7898
		7898	7898	7898	7898	7898/M	7898/M	7898/M	7898
		7898/P	7979	7979	7979	7979	7979/S	7979	7979
		7979	7979	7979	7979/M	7979/M	7979/M	7979	7979/P
		8004	8016	8091	8172	8172	8172	8172	8172/S
		8172	8172	8172	8172	8172	8172/M	8172/M	8172/M
		8172	8172/P	8227	8227	8227	8227	8227/S	8227
		8227	8227	8227	8227	8227/M	8227/M	8227/M	8227
		8227/P	8499	8499	8499	8499	8499/S	8499	8499
		8499	8499	8499	8499/M	8499/M	8499/M	8499	8499/P
		8630	8630	8630	8630	8630/S	8630	8630	8630
		8630	8630	8630/M	8630/M	8630/M	8630	8630/P	9406
		9406	9406	9406	9406/S	9406	9406	9406	9406
		9406	9406/M	9406/M	9406/M	9406	9406/P	9479	9479
		9479	9479	9479/S	9479	9479	9479	9479	9479
		9479/M	9479/M	9479/M	9479	9479/P	9623	9623	9623
		9623	9623/S	9623	9623	9623	9623	9623	9623/M
		9623/M	9623/M	9623	9623/P	9647	9647	9647	9647
		9647/S	9647	9647	9647	9647	9647	9647/M	9647/M
		9647/M	9647	9647/P	10201	10201	10201	10201	10201/S
		10201	10201	10201	10201	10201	10201/M	10201/M	10201/M
		10201	10201/P	10519	10519	10519	10519	10519/S	10519
		10519	10519	10519	10519	10519/M	10519/M	10519/M	10519
		10519/P	10550	10550	10550	10550	10550/S	10550	10550
		10550	10550	10550	10550/M	10550/M	10550/M	10550	10550/P
		10579	10579	10579	10579	10579/S	10579	10579	10579
		10579	10579	10579/M	10579/M	10579/M	10579	10579/P	10925
		10925	10925	10925	10925/S	10925	10925	10925	10925
		10925	10925/M	10925/M	10925/M	10925	10925/P	11139	11139
		11139	11139	11139/S	11139	11139	11139	11139	11139
		11139/M	11139/M	11139/M	11139	11139/P	11151	11151	11151
		11151	11151/S	11151	11151	11151	11151	11151	11151/M
		11151/M	11151/M	11151	11151/P				
mtv$mli_status	5535	7904/M	9008	9013/M	10451	10451			
mtv$monitor_segment_table	5923	8447/M							
mtv$scb	5771	10240	10241						
multiprocessing_allowed	1284	7865	7897	7977	8171	8498	8627	9404	9478
		9622	9646	10200	10550	10578	10641	10686	10924
nat$received_message_descriptor	3015	3008	3017						
nat$received_message_list	3007	2524							
new_entry_status	5359	5366/S	5367/S	5369	5372	5376			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
new_entry_status	8381	8471/S	8471/S	8471	8471	8471		
new_entry_status	9936	10081/S	10081/S	10081	10081	10081		
new_max_ptlo	9946	10105/M	10107	10110/S	10118			
new_scheduling_priority	8777	8838/M	8843	8844/M	8848			
new_scheduling_priority	8892	8916/M	8920	8921/M	8925			
new_task_status	4936	7749	7756/M	7849	7850/M	7964	7967/M	8018	8019
		8022/M	8078	8102/M	8152	8152	8157	8160/M	8929/M
		9034/M	9036/M	9038/M	9042/M	9044/M	9135/M	9153	9181/M
		10480	10556	10557	10558/M	11119	11120	11121/M	
new_task_status	9090	9114/M	9116/M	9118/M	9122/M	9124/M	9135	9136	9137
		9138	9153/M	9154	9161/M	9164/M	9167		
next_avail_ptlo	6594	6602/M	6603	6607/S	6634				
next_avail_ptlo	8381	8445/M	8445	8445/S	8445				
next_avail_ptlo	8581	8608/M	8608	8608/S	8608				
next_cyclic_aging_time	2258	8754	8754/M	8754/M	9303	9305/M	9307/M	9435	9437/M
		9439/M	9888	9890/M	9892/M	10530	10530/M	10530/M	
next_ijle_p	7650	7694/M	7695	7698/P	7701	7702	7703		
next_index	8276	8288/M	8288/S	8288	8288	8288	8288/M	8288/S	8288/S
		8288	8288	8288	8288	8288	8288/S		
next_index	8315	8319/M	8327/S	8327/S	8327	8327	8328	8328	8329
		8330/M	8330/S	8333	8337	8338	8347/S		
next_index	10370	10379/M	10381/M	10386	10386/S	10388/S	10389/S	10391	10396/S
		10397/M	10397	10400/M					
next_index	10408	10561/M	10561/S	10561	10561	10561	10561/M	10561/S	10561/S
		10561	10561	10561	10561	10561	10561/S		
next_index	11072	11077/M	11088						
next_index	11095	11124/M	11124/S	11124	11124	11124	11124/M	11124/S	11124/S
		11124	11124	11124	11124	11124	11124/S		
next_processor_state	1150	10481	10594						
next_ptlo	4832	7666/M	7671/M	7674/M	7683	7701/M	7714	7721/S	7723/M
		7723/S							
next_ptlo_to_dispatch	1182	6960	6965/S	6972	6973	6974	6975	6982	6987/M
		7759	7866	7866/S	7866	7866	7866	7866	7866
		7866/M	7898	7898/S	7898	7898	7898	7898	7898
		7898/M	7979	7979/S	7979	7979	7979	7979	7979
		7979/M	8025	8093	8172	8172/S	8172	8172	8172
		8172	8172	8172/M	8227	8227/S	8227	8227	8227
		8227	8227	8227/M	8499	8499/S	8499	8499	8499
		8499	8499	8499/M	8560	8630	8630/S	8630	8630
		8630	8630	8630	8630/M	8747	8816	8826	8934
		9050	9175	9406	9406/S	9406	9406	9406	9406
		9406	9406/M	9479	9479/S	9479	9479	9479	9479
		9479	9479/M	9623	9623/S	9623	9623	9623	9623
		9623	9623/M	9647	9647/S	9647	9647	9647	9647
		9647	9647/M	10201	10201/S	10201	10201	10201	10201
		10201	10201/M	10510	10511	10517/M	10519	10519/S	10519
		10519	10519	10519	10519	10519/M	10540	10550	10550/S
		10550	10550	10550	10550	10550	10550/M	10577	10579
		10579/S	10579	10579	10579	10579	10579	10579/M	10595
		10631	10632	10634/M	10925	10925/S	10925	10925	10925
		10925	10925	10925/M	11139	11139/S	11139	11139	11139
		11139	11139	11139/M	11145	11146/S	11148/S	11149/P	11151/P
		11151	11151/S	11151	11151	11151	11151	11151	11151/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES				
		11152/M				
next_ptlo_to_dispatch	10425	10511/M	10512/S	10514/S	10515/P	10518/P
next_ready_time	9222	9233/M	9261	9262/M	9267	
next_search_ptlo	6863	7054/M	7054			
next_search_ptlo	6863	7056/M	7056			
next_search_ptlo	6863	7058/M	7058			
next_search_ptlo	7082	7104/M	7123			
next_search_ptlo	7194	7216/M	7238			
next_search_ptlo	7302	7318/M	7344			
next_search_ptlo	7766	7866/M	7866	7898/M	7898	
next_search_ptlo	7766	7866/M	7866	7898/M	7898	
next_search_ptlo	7766	7866/M	7866	7898/M	7898	
next_search_ptlo	7932	7979/M	7979			
next_search_ptlo	7932	7979/M	7979			
next_search_ptlo	7932	7979/M	7979			
next_search_ptlo	8110	8172/M	8172			
next_search_ptlo	8110	8172/M	8172			
next_search_ptlo	8110	8172/M	8172			
next_search_ptlo	8182	8227/M	8227			
next_search_ptlo	8182	8227/M	8227			
next_search_ptlo	8182	8227/M	8227			
next_search_ptlo	8381	8499/M	8499			
next_search_ptlo	8381	8499/M	8499			
next_search_ptlo	8381	8499/M	8499			
next_search_ptlo	8581	8630/M	8630			
next_search_ptlo	8581	8630/M	8630			
next_search_ptlo	8581	8630/M	8630			
next_search_ptlo	9349	9406/M	9406			
next_search_ptlo	9349	9406/M	9406			
next_search_ptlo	9349	9406/M	9406			
next_search_ptlo	9446	9479/M	9479			
next_search_ptlo	9446	9479/M	9479			
next_search_ptlo	9446	9479/M	9479			
next_search_ptlo	9584	9623/M	9623	9647/M	9647	
next_search_ptlo	9584	9623/M	9623	9647/M	9647	
next_search_ptlo	9584	9623/M	9623	9647/M	9647	
next_search_ptlo	10157	10201/M	10201			
next_search_ptlo	10157	10201/M	10201			
next_search_ptlo	10157	10201/M	10201			
next_search_ptlo	10408	10519/M	10519	10550/M	10550	10579/M 10579
next_search_ptlo	10408	10519/M	10519	10550/M	10550	10579/M 10579
next_search_ptlo	10408	10519/M	10519	10550/M	10550	10579/M 10579
next_search_ptlo	10879	10925/M	10925			
next_search_ptlo	10879	10925/M	10925			
next_search_ptlo	10879	10925/M	10925			
next_search_ptlo	11095	11139/M	11139	11151/M	11151	
next_search_ptlo	11095	11139/M	11139	11151/M	11151	
next_search_ptlo	11095	11139/M	11139	11151/M	11151	
next_state	2489	10480	10587			
next_task_ijle_p	10426	10512/P	10513/M	10513	10515/P	10518/P
next_task_ijle_p	11099	11146/P	11147/M	11147	11149/P	11151/P
next_task_xcb_p	10427	10516/P	10518/P			
next_task_xcb_p	11100	11150/P	11151/P			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
nlc$cc_connect_confirm	3047	3038							
nlc$cc_connect_request	3046	3036							
nlc$cc_expedited_data	3052	3038							
nlc$cc_max_pdu_kind	3054	3057							
nlc$channel_connection_pdu	3070	3022							
nlc$channelnet_pdu	3070	3024							
nlt$cc_pdu_kind	3057	3035							
nlt$cc_seq#_or_connect_time	3034	3023							
nlt$cc_sequence_number	3060	3039							
nlt$device_identifier	3067	3018							
nlt$pdu_type	3070	3021							
normal	274	5650/M	6489/M	6496/M	6529/M	6534/M	6601/M	6604/M	7449/M
		7452/M	7452/M	7453	7461/M	8429	8445/M	8445/M	8446
		8608/M	8608/M	8609	8702/M	8704/M	8718	9102/M	9102/M
		9103	9334	9365/M	9373/M	9412/M	9433/M	9502/M	9543/M
		9577/M	9595/M	9920/M	9950/M	9976/M	10978/M	10986/M	
null_debug_mask	10838	10842							
null_dispatching_info	7799	7818							
null_pva	6326	8448							
offset	1999	7595	8006/M	8006					
offset	5243	5248/M	5255/M	5258					
offset	5286	5293/M	5293/M	5293					
offset	8182	8222/M	8222/M	8222	8232/M	8232/M	8232		
offset	8601	8605/M	8606						
old_dispatching_priority	7797	7819/M	7822						
old_entry_status	5362	5364/M	5366/S	5367/S	5371	5375			
old_entry_status	8381	8471/M	8471/S	8471/S	8471	8471			
old_entry_status	9936	10081/M	10081/S	10081/S	10081	10081			
old_max_ptlo	9947	10104/M	10107	10114	10116				
old_te	5579	5594/M							
old_te	5617	5630							
operator_set_dispatching_prio	1452	9994/M	10000/M	10031/M	10323	10325			
option	6466	6495							
option	6507	6533							
option	7416	7452							
option	9057	9102							
osc$aging_interval_maximum	2333	2336							
osc$base_exception	5036	5038	6094						
osc$call_instruction	3250	3258							
osc$cpu_stepped	2492	10480	10587						
osc$data_read	3249	3258							
osc$debug	3087	10845	10846						
osc$free_flag	3085	9022	9156	10792	10850				
osc$free_running_clock_maximum	1524	1521							
osc$idle_no_io_active	2484	10739							
osc$idle_with_io_active	2483	10737	10767						
osc$invalid_ring	1945	1985							
osc$keypoint_base	6094	6096	6100	6104	6107	6111	6115	6119	6122
		6126	6130	6134	6138	6141	6144	6148	6151
		6155	6158	6163					
osc$keypoint_enable	3155	8484	8486	8669	8671				
osc$max_fault_contents	3450	3444							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$max_idle_count	2471	2479							
osc$max_kpt_pages	5962	5968	5970	5972	6008				
osc$max_name_size	2299	2303	2306						
osc$max_number_of_processors	716	1140	5931	5937	5997	5998			
osc$max_page_frames	1895	1332	1333	1814	1816	2052	2093	2201	2207
osc$max_page_size	2432	2428							
osc$max_page_table_entries	1896	1899							
osc$max_ring	1944	1985	1986						
osc$max_segment_length	1968	1991	2744	2775					
osc$max_status_condition_code	240	236	252						
osc$max_status_condition_number	5655	5646							
osc$max_string_size	256	259	262	267					
osc$max_tasks	110	107	9361	9500					
osc$maximum_offset	1967	1968	1988	1988	1989				
osc$maximum_processor_id	3275	3271							
osc$maximum_processor_number	2455	2450							
osc$maximum_processors	720	716	2455						
osc$maximum_segment	1966	1987							
osc$min_ecc	5035	5036							
osc$min_page_size	2431	2428							
osc$min_ring	1943	1986							
osc$not_idle	2483	10734	10765	10775					
osc$pr_base_constant	3811	6428	6447	6490	6494	6502	7450	7454	7466
		7501	7512	7529	7547	7573	7599	7653	7725
		7748	7757	7806	7927	7956	7983	8017	8039
		8070	8104	8134	8176	8265	8270	8282	8292
		8444	8451	8508	8532	8553	8607	8610	8677
		8727	8756	8783	8870	8897	8961	9004	9010
		9028	9048	9100	9104	9211	9247	9250	9257
		9271	9329	9339	9469	9486	9504	9539	9573
		9607	9631	9655	9951	9975	10137	10141	10170
		10210	10233	10246	10461	10523	10589	10593	10820
		10981	10988	11022	11107	11155			
osc$pr_element_id	3800	4883							
osc$pr_process_interval_timer	3837	4884	8732	9922	9955				
osc$pr_set_keypoint_enable	5948	10809							
osc$pr_user_mask_reg	3846	4883							
osc$purge_all_cache	3852	10644	10649	10689	10695				
osc$purge_all_page_seg_map	3861	8448	10645	10650	10690	10696			
osc$system_keypoints	6040	8478	8652						
osc$system_sample_keypoints	6041	8479	8653						
osc$system_table_lock_set	728	6910	7862	7866	7898	7979	8172	8227	8499
		8630	9386	9406	9479	9519	9553	9623	9647
		10201	10482	10519	10550	10579	10925	11139	11151
osc$task_time_slice_maximum	1513	1516							
osc$tsrv_ring	1948	9168							
osc$vl_invalid_entry	2694	8447							
osk$base	833	735	739	743	747	751	755	759	763
		767	771	775	779	783	787	792	795
		798							
osk$m	904	8474	8475	10524	10822				
osk$monitor_multiplier	903	10441	10524	10822					
osk$mtr	868	10441	10524	10822					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osk$performance	869	8474	8475						
osk$system_class	879	863	864	865	866	867	868	869	
osp$update_job_keypoint_mask	5659	9461	9599						
ost$aging_interval	2336	2255	2256						
ost$asid	1931	1276	1927	2071	2082	2083	2098	2700	2837
		5460	5731						
ost$binary_unique_name	4043	3878							
ost$class_15_keypoint	6058	6015							
ost$cp_time	1629	1597	1642	2541	4781				
ost$cp_time_value	1627	1318	1630	1631	2250	2251	2554		
ost$cpu_element_id	2447	1171							
ost$cpu_idle_statistics	2474	1174							
ost$cpu_memory_port_mask	2449	1147	5528						
ost$cpu_running_or_stepped	2492	2489	2489						
ost$cpu_state	2487	1156							
ost$cpu_state_reason	2498	1177							
ost$cpu_state_table	1143	1140	5633	6871	7354	7377	7568	7744	7796
		8002	8014	8064	8383	8515	8583	8684	8771
		8877	8968	9059	9451	9662	9875	9902	9938
		10410	10966	11096					
ost$cs_lock	3289	2522							
ost$cst_trace_control	3685	1175							
ost$date_time	6077	6060							
ost$debug_code	3249	3237							
ost$debug_list	3245	3149							
ost$debug_list_entry	3236	3245							
ost$debug_mask	3255	3148	4755	10838	10838				
ost$exchange_package	3098	2509							
ost$execute_privilege	2713	2695	2708						
ost$execution_control_block	2508	1157	2534	4987	5006	5540	5541	5547	5554
		5555	5561	5567	5764	6568	6864	7085	7197
		7305	7418	7474	7519	7571	7610	7623	7802
		7952	8130	8214	8415	8529	8602	8762	8773
		8781	8889	9224	9363	9454	9497	9592	10166
		10230	10427	10434	10835	10889	11100	11103	
ost$external_interrupt_request	3673	1163							
ost$family_name	2346	2341							
ost$flags	3155	3105	8484	8486	8669	8671			
ost$frame_descriptor	3213	3228							
ost$free_running_clock	1521	1287	1288	1289	1290	1324	1334	1335	1336
		1453	1465	2097	2249	2258	2540	3894	4692
		4796	4803	4804	4844	4845	4857	4858	4905
		6189	6190	9032	9112				
ost$global_task_id	101	96	583	1152	1281	1310	2245	2519	2520
		3334	3537	3896	3984	4859	4979	4988	4998
		5008	5016	5024	5744	5751	5757	5918	5918
		6353	6353	6380	6465	6506	6545	6559	6570
		7353	7376	7417	7473	7518	7563	7745	7767
		7933	8045	8112	8183	8260	10956	10971	
ost$halfword	2004	2460							
ost$idle_type	2483	2478							
ost$key_lock	1974	2701	2822						
ost$key_lock_value	1980	1977	3172	3174					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$keypoint	6049	6059							
ost$keypoint_class	3187	3118	3189						
ost$keypoint_control	5979	5942							
ost$keypoint_environment	6040	5986							
ost$keypoint_mask	3189	3121	5985	5985	5985	5985	8417	8477	8483
		8599	8664	8668					
ost$keypoint_multipro_option	6044	5987							
ost$logical_processor_id	2450	1148							
ost$minimum_save_area	3223	3110	3198	3431					
ost$monitor_condition	3074	3081							
ost$monitor_conditions	3081	3111	3115	3203	3506	3520			
ost$monitor_fault	3427	3376							
ost$monitor_fault_contents	3444	3440							
ost$name	2306	2295	2344	2346	2568	2632	3469	6327	6327
		6328							
ost$p_register	3170	3099	3224	3498	3504				
ost$page_id	1901	1911							
ost$page_size	2428	2409							
ost$page_table_entry	1906	1915	2068						
ost$page_table_index	1899	1915	2137						
ost$paging_statistics	1665	1643	2549						
ost$parcel	2006	1169	1170						
ost$pre_processed_for_reconfig	3681	1178							
ost$processor_element_id	2459	2447							
ost$processor_element_number	2468	2461							
ost$processor_id	3271	2512	3265						
ost$processor_id_set	3265	2511	5781	5782	5903	5999	10240		
ost$processor_keypoint_control	5966	6001							
ost$processor_model_number	2358	2352	2462	4045					
ost$processor_serial_number	2436	2353	2463	4044					
ost$pva	1996	3143	3161	3175	3428	3521	6383		
ost$read_privilege	2716	2696	2709						
ost$real_memory_address	1919	1168							
ost$register_number	3166	3140	3209	3217	3218	3219			
ost$ring	1985	1997	2698	2699	2733	2813	2814	3160	
ost$ring1_termination_reason	3283	2545							
ost$segment	1987	1998	2816	2946	3138	3239	3895	4970	5553
		5568	8212	8411	8412				
ost$segment_access_control	2706	2835							
ost$segment_descriptor	2693	2667							
ost$segment_length	1991	2818	2820	2842	2844	2847			
ost$segment_offset	1988	1928	1999	2748	3240	3242			
ost$stack_frame_save_area	3197	3231	3464						
ost$state_tables	1140	5639							
ost$status	224	588	2829	3499					
ost$status_condition	248	275	606						
ost$status_condition_code	252	227	248						
ost$string	265	228							
ost$string_size	259	266							
ost$system_flag	3605	3601	5745						
ost$system_virtual_address	1926	2142							
ost$task_index	107	102	1182	2170	2171	4832	4931	4944	6381
		6594	6646	6661	6677	6678	6731	6747	6748

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		6866	7070	7081	7082	7083	7084	7182	7193
		7194	7195	7196	7291	7301	7302	7303	7304
		7394	7394	7608	8280	8299	8315	8316	8778
		8893	8894	9223	9283	9356	9359	9455	9496
		9590	9880	9946	9947	10164	10425	10436	10833
		10866	10890	10973	10973	11072	11073	11102	
ost$task_time_slice	1516	1502							
ost$top_of_stack_pointer	3158	3150							
ost$trap_enable	3192	3107	3495						
ost$user_condition	3084	3091	4756						
ost$user_conditions	3091	3109	3113	3201	3230	3467	3507	9022	9156
		10792	10846	10850					
ost$user_identification	2339	2244							
ost$user_name	2344	2340							
ost$valid_relative_pointer	1994	2538	2539	3901	3904				
ost$valid_ring	1986	3150	7564						
ost$vector_simulation_control	5901	5783							
ost$virtual_machine_identifier	3180	3101	3103	3225					
ost$wait	4731	4714	5007						
ost$write_privilege	2719	2697	2710						
ost$x_register	3167	3140	3209						
osv$cpus_logically_on	5931	6427	6446	6490	6494	6502	7450	7454	7466
		7501	7512	7529	7547	7573	7599	7653	7725
		7748	7757	7806	7927	7956	7983	8017	8039
		8070	8104	8134	8176	8265	8270	8282	8292
		8444	8451	8508	8532	8553	8607	8610	8677
		8727	8756	8783	8870	8897	8961	9004	9010
		9028	9048	9100	9104	9211	9247	9250	9257
		9271	9329	9339	9469	9486	9504	9539	9573
		9607	9631	9655	9951	9975	10137	10141	10170
		10210	10233	10246	10461	10523	10589	10593	10640
		10684	10820	10981	10988	11022	11107	11155	
osv$cpus_physically_configured	5937	6942	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10519	10550	10579
		10925	11139	11151					
osv$keypoint_control	5942	8478	8479	8480	8652	8653	8654	8660	
osv$special_aam_trap	6362	8724							
osv$trap_task_errors	8698	8708							
p_register	3099	7590	7592	8006/M	8006	9007	9168		
parent_global_task_id	2520	8647/M	8703						
parent_global_task_id	5024	8703	8717/P						
pit_count	2540	9962/M	9962						
pit_value	4885	9961	9962	9963	9964				
pit_value	8699	8733/M	8735	8736/M	8736	8738			
pmc$kill_task_flag	3605	3621							
pmc$max_signal_contents	3588	3582							
pmc$max_task_id	3343	3340							
pmt$binary_mainframe_id	2351	2248							
pmt$condition_identifier	3417	3411							
pmt$cpu_model_number	2418	2407	2414						
pmt$cpu_serial_number	2421	2408	2413						
pmt$initialization_value	2891	2826	3893						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pmt$sense_switches	2441	2259							
pmt$signal	3544	3538	5025	5758					
pmt$signal_contents	3582	3546							
pmt$signal_id	3549	3545							
pmt$task_id	3340	2536	3335						
pmt$vector_simulation	5908	5902							
preselect_loop	6944	6944	6950	6957					
preselect_loop	7766	7866	7866	7866	7898	7898	7898		
preselect_loop	7932	7979	7979	7979					
preselect_loop	8110	8172	8172	8172					
preselect_loop	8182	8227	8227	8227					
preselect_loop	8381	8499	8499	8499					
preselect_loop	8581	8630	8630	8630					
preselect_loop	9349	9406	9406	9406					
preselect_loop	9446	9479	9479	9479					
preselect_loop	9584	9623	9623	9623	9647	9647	9647		
preselect_loop	10157	10201	10201	10201					
preselect_loop	10408	10519	10519	10519	10550	10550	10550	10579	10579
		10579							
preselect_loop	10879	10925	10925	10925					
preselect_loop	11095	11139	11139	11139	11151	11151	11151		
previous_index	8276	8288/M	8288/M	8288/S	8288	8288			
previous_index	8316	8320/M	8329/M	8341/S	8343	8344			
previous_index	10408	10561/M	10561/M	10561/S	10561	10561			
previous_index	11095	11124/M	11124/M	11124/S	11124	11124			
priority	1322	9201	9202	9202	9203/M	9203	9205	9206/M	9206
priority	10428	10669/M	10673	10673	10674/S	10711	10713	10715	10720/M
priority_mask	6402	6414							
process_interval_timer_1	3127	8733	9924	9956	9963/M				
process_interval_timer_2	3129	8734	9925	9957	9964/M				
processor_selections	2511	6948	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10240	10241/M	10519
		10550	10579	10685	10925	11139	11151		
processor_state	1149	10594							
processors_logically_on	5781	10240	10241						
pseudo_rb	7800	7813/M	7814/M	7815/M	7816/M	7817/M	7818/M	7821/P	
ptk$new_job_name_1	935	8474							
ptk$new_job_name_2	938	8475							
ptk$performance_base	809	908	911	914	917	920	923	926	929
		932	935	938	941	944	947	950	953
		956	959	962	965	968	971	974	977
		980	983	986	989	992	995	998	1001
		1004	1007	1010	1013	1016	1019	1022	1025
		1028	1031	1034	1037	1040	1043	1046	1049
		1052	1055	1058	1061	1064	1067	1070	1073
		1076	1079	1082	1085	1088	1091	1094	1097
		1100	1103	1106	1109	1112	1115	1118	1121
		1124	1127						
ptl_flags	4941	7836	7922	7923	7924/M	8850/M	8927/M	9143	9163
		9169/M	9171/M	9180/M	9426/M	10801	10804/M	10848	
ptl_lock_set	5669	5671/M							
ptl_p	4893	10105	10106/P	10108/M	10110/M	10111			
ptl_p	6595	6607/M	6608	6614/M	6616/M	6623/M	6624/M	6625/M	6626/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		6627/M	6627	6628/M	6629/M	6630/M	6631/M	6632/M	6633/M
		6635							
ptl_p	7801	7808/M	7809/P	7810	7816	7836	7837	7838	7839/M
		7848	7848	7849	7850/M	7853	7869	7876	7877/P
		7878	7882	7882	7888	7890/M	7891/M	7892/P	7893
		7895/M	7909	7910/P	7913	7914/P	7922	7923	7924/M
ptl_p	7953	7958/M	7959/P	7963	7964	7964	7967/M	7969	7971/M
		7972/P	7973/M	7974	7976/M	7978			
ptl_p	8065	8068/M	8074	8074	8078	8087/M	8102/M		
ptl_p	8131	8136/M	8137	8140/P	8146	8148/M	8151	8151	8152
		8152	8156	8157	8157	8160/M	8162	8164/M	8165/M
		8166/P	8167	8169/M	8170				
ptl_p	8185	8217	8218	8223/P	8225/M	8249/P	8251/M	8252/M	8253/P
ptl_p	8263	8267/M	8268	8269/S					
ptl_p	8381	8445/M	8445	8445/M	8445/M	8445/M	8445/M	8445/M	8445/M
		8445/M	8445	8445/M	8445/M	8445/M	8445/M	8445/M	8445/M
		8445							
ptl_p	8581	8608/M	8608	8608/M	8608/M	8608/M	8608/M	8608/M	8608/M
		8608/M	8608	8608/M	8608/M	8608/M	8608/M	8608/M	8608/M
		8608							
ptl_p	10975	10998/M	10999/M	11000/M	11001/M	11002/M	11003/M	11004/M	11005/M
		11006/M	11007/M	11008/M	11009/M	11010/M	11019		
ptl_thread	4931	6608	6650/M	6654/M	6696	6698	6701	6707	6712/M
		6712	6723/M	6761	6766	6771/M	6771	6971/M	6981/M
		7007/M	7020/M	7020	7021/M	7030	7034/M	7034	7035/M
		7037	7041/M	7042/M	7054	7054	7054/M	7054	7054/M
		7054	7054/M	7054	7054	7054	7054	7054	7054/M
		7054	7054/M	7054	7054/M	7054	7056	7056	7056/M
		7056	7056	7056/M	7056	7056/M	7056	7056	7056
		7056	7056	7056	7056/M	7056	7056/M	7056	7056/M
		7056	7058	7058	7058	7058	7058/M	7058	7058/M
		7058/M	7104	7110	7112/M	7112	7114/M	7114	7115/M
		7133	7146	7155	7160	7163	7168/M	7168	7169/M
		7169	7170/M	7171	7216	7222	7224/M	7224	7227
		7229/M	7229	7230/M	7243	7255	7265	7269	7272
		7275	7277/M	7277	7278/M	7278	7279/M	7280	7318
		7324	7327	7330	7333/M	7333	7335/M	7336/M	7864
		7864	7864	7864	7864/M	7864	7864/M	7866/M	7866/M
		7866/M	7866/M	7866	7866/M	7866	7866/M	7866	7866/M
		7866	7866/M	7866/M	7866	7866	7866/M	7866	7866/M
		7866	7866/M	7866	7866	7866	7866	7866	7866/M
		7866	7866/M	7866	7866/M	7866	7866	7866	7866/M
		7866	7866	7866/M	7866	7866/M	7866	7866	7866
		7866	7866	7866	7866/M	7866	7866/M	7866	7866/M
		7866	7866	7866	7866	7866	7866/M	7866	7866/M
		7866/M	7898/M	7898/M	7898/M	7898/M	7898	7898/M	7898
		7898/M	7898	7898/M	7898	7898/M	7898/M	7898	7898
		7898/M	7898	7898/M	7898	7898/M	7898	7898	7898
		7898	7898	7898/M	7898	7898/M	7898	7898/M	7898
		7898	7898	7898/M	7898	7898	7898/M	7898	7898/M
		7898	7898	7898	7898	7898	7898	7898/M	7898
		7898/M	7898	7898/M	7898	7898	7898	7898	7898
		7898/M	7898	7898/M	7898/M	7979/M	7979/M	7979/M	7979/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		7979	7979/M	7979	7979/M	7979	7979/M	7979	7979/M
		7979/M	7979	7979	7979/M	7979	7979/M	7979	7979/M
		7979	7979	7979	7979	7979	7979/M	7979	7979/M
		7979	7979/M	7979	7979	7979	7979/M	7979	7979
		7979/M	7979	7979/M	7979	7979	7979	7979	7979
		7979	7979/M	7979	7979/M	7979	7979/M	7979	7979
		7979	7979	7979	7979/M	7979	7979/M	7979/M	8172/M
		8172/M	8172/M	8172/M	8172	8172/M	8172	8172/M	8172
		8172/M	8172	8172/M	8172/M	8172	8172	8172/M	8172
		8172/M	8172	8172/M	8172	8172	8172	8172	8172
		8172/M	8172	8172/M	8172	8172/M	8172	8172	8172
		8172/M	8172	8172	8172/M	8172	8172/M	8172	8172
		8172	8172	8172	8172	8172/M	8172	8172/M	8172
		8172/M	8172	8172	8172	8172	8172	8172/M	8172
		8172/M	8172/M	8227/M	8227/M	8227/M	8227/M	8227	8227/M
		8227	8227/M	8227	8227/M	8227	8227/M	8227/M	8227
		8227	8227/M	8227	8227/M	8227	8227/M	8227	8227
		8227	8227	8227	8227/M	8227	8227/M	8227	8227/M
		8227	8227	8227	8227/M	8227	8227	8227/M	8227
		8227/M	8227	8227	8227	8227	8227	8227	8227/M
		8227	8227/M	8227	8227/M	8227	8227	8227	8227
		8227	8227/M	8227	8227/M	8227/M	8445	8499/M	8499/M
		8499/M	8499/M	8499	8499/M	8499	8499/M	8499	8499/M
		8499	8499/M	8499/M	8499	8499	8499/M	8499	8499/M
		8499	8499/M	8499	8499	8499	8499	8499	8499/M
		8499	8499/M	8499	8499/M	8499	8499	8499	8499/M
		8499	8499	8499/M	8499	8499/M	8499	8499	8499
		8499	8499	8499	8499/M	8499	8499/M	8499	8499/M
		8499	8499	8499	8499	8499	8499/M	8499	8499/M
		8499/M	8544/M	8544/M	8608	8630/M	8630/M	8630/M	8630/M
		8630	8630/M	8630	8630/M	8630	8630/M	8630	8630/M
		8630/M	8630	8630	8630/M	8630	8630/M	8630	8630/M
		8630	8630	8630	8630	8630	8630/M	8630	8630/M
		8630	8630/M	8630	8630	8630	8630/M	8630	8630
		8630/M	8630	8630/M	8630	8630	8630	8630	8630
		8630	8630/M	8630	8630/M	8630	8630/M	8630	8630
		8630	8630	8630	8630/M	8630	8630/M	8630/M	8741/M
		8741/M	9406/M	9406/M	9406/M	9406/M	9406	9406/M	9406
		9406/M	9406	9406/M	9406	9406/M	9406/M	9406	9406
		9406/M	9406	9406/M	9406	9406/M	9406	9406	9406
		9406	9406	9406/M	9406	9406/M	9406	9406/M	9406
		9406	9406	9406/M	9406	9406	9406/M	9406	9406/M
		9406	9406	9406	9406	9406	9406	9406/M	9406
		9406/M	9406	9406/M	9406	9406	9406	9406	9406
		9406/M	9406	9406/M	9406/M	9417	9417	9417	9417
		9417/M	9417	9417/M	9479/M	9479/M	9479/M	9479/M	9479
		9479/M	9479	9479/M	9479	9479/M	9479	9479/M	9479/M
		9479	9479	9479/M	9479	9479/M	9479	9479/M	9479
		9479	9479	9479	9479	9479/M	9479	9479/M	9479
		9479/M	9479	9479	9479	9479/M	9479	9479	9479/M
		9479	9479/M	9479	9479	9479	9479	9479	9479
		9479/M	9479	9479/M	9479	9479/M	9479	9479	9479
		9479	9479	9479/M	9479	9479/M	9479/M	9528	9528

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		9528	9528	9528/M	9528	9528/M	9562	9562	9562
		9562	9562/M	9562	9562/M	9623/M	9623/M	9623/M	9623/M
		9623	9623/M	9623	9623/M	9623	9623/M	9623	9623/M
		9623/M	9623	9623	9623/M	9623	9623/M	9623	9623/M
		9623	9623	9623	9623	9623	9623/M	9623	9623/M
		9623	9623/M	9623	9623	9623	9623/M	9623	9623
		9623/M	9623	9623/M	9623	9623	9623	9623	9623
		9623	9623/M	9623	9623/M	9623	9623/M	9623	9623
		9623	9623	9623	9623/M	9623	9623/M	9623/M	9647/M
		9647/M	9647/M	9647/M	9647	9647/M	9647	9647/M	9647
		9647/M	9647	9647/M	9647/M	9647	9647	9647/M	9647
		9647/M	9647	9647/M	9647	9647	9647	9647	9647
		9647/M	9647	9647/M	9647	9647/M	9647	9647	9647
		9647/M	9647	9647	9647/M	9647	9647/M	9647	9647
		9647	9647	9647	9647	9647/M	9647	9647/M	9647
		9647/M	9647	9647	9647	9647	9647	9647/M	9647
		9647/M	9647/M	10108/M	10110/M	10116/M	10198	10198	10198
		10198	10198/M	10198	10198/M	10201/M	10201/M	10201/M	10201/M
		10201	10201/M	10201	10201/M	10201	10201/M	10201	10201/M
		10201/M	10201	10201	10201/M	10201	10201/M	10201	10201/M
		10201	10201	10201	10201	10201	10201/M	10201	10201/M
		10201	10201/M	10201	10201	10201	10201/M	10201	10201
		10201/M	10201	10201/M	10201	10201	10201	10201	10201
		10201	10201/M	10201	10201/M	10201	10201/M	10201	10201
		10201	10201	10201	10201/M	10201	10201/M	10201/M	10519/M
		10519/M	10519/M	10519/M	10519	10519/M	10519	10519/M	10519
		10519/M	10519	10519/M	10519/M	10519	10519	10519/M	10519
		10519/M	10519	10519/M	10519	10519	10519	10519	10519
		10519/M	10519	10519/M	10519	10519/M	10519	10519	10519
		10519/M	10519	10519	10519/M	10519	10519/M	10519	10519
		10519	10519	10519	10519	10519/M	10519	10519/M	10519
		10519/M	10519	10519	10519	10519	10519	10519/M	10519
		10519/M	10519/M	10550/M	10550/M	10550/M	10550/M	10550	10550/M
		10550	10550/M	10550	10550/M	10550	10550/M	10550/M	10550
		10550	10550/M	10550	10550/M	10550	10550/M	10550	10550
		10550	10550	10550	10550/M	10550	10550/M	10550	10550/M
		10550	10550	10550	10550/M	10550	10550	10550/M	10550
		10550/M	10550	10550	10550	10550	10550	10550	10550/M
		10550	10550/M	10550	10550/M	10550	10550	10550	10550
		10550	10550/M	10550	10550/M	10550/M	10579/M	10579/M	10579/M
		10579/M	10579	10579/M	10579	10579/M	10579	10579/M	10579
		10579/M	10579/M	10579	10579	10579/M	10579	10579/M	10579
		10579/M	10579	10579	10579	10579	10579	10579/M	10579
		10579/M	10579	10579/M	10579	10579	10579	10579/M	10579
		10579	10579/M	10579	10579/M	10579	10579	10579	10579
		10579	10579	10579/M	10579	10579/M	10579	10579/M	10579
		10579	10579	10579	10579	10579/M	10579	10579/M	10579/M
		10704	10725	10725	10725	10725	10725/M	10725	10725/M
		10925/M	10925/M	10925/M	10925/M	10925	10925/M	10925	10925/M
		10925	10925/M	10925	10925/M	10925/M	10925	10925	10925/M
		10925	10925/M	10925	10925/M	10925	10925	10925	10925
		10925	10925/M	10925	10925/M	10925	10925/M	10925	10925
		10925	10925/M	10925	10925	10925/M	10925	10925/M	10925

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10925	10925	10925	10925	10925	10925/M	10925	10925/M
		10925	10925/M	10925	10925	10925	10925	10925	10925/M
		10925	10925/M	10925/M	10997	10997	10997/M	10997	10999/M
		11078	11086/M	11086/M	11139/M	11139/M	11139/M	11139/M	11139
		11139/M	11139	11139/M	11139	11139/M	11139	11139/M	11139/M
		11139	11139	11139/M	11139	11139/M	11139	11139/M	11139
		11139	11139	11139	11139	11139/M	11139	11139/M	11139
		11139/M	11139	11139	11139	11139/M	11139	11139	11139/M
		11139	11139/M	11139	11139	11139	11139	11139	11139
		11139/M	11139	11139/M	11139	11139/M	11139	11139	11139
		11139	11139	11139/M	11139	11139/M	11139/M	11151/M	11151/M
		11151/M	11151/M	11151	11151/M	11151	11151/M	11151	11151/M
		11151	11151/M	11151/M	11151	11151	11151/M	11151	11151/M
		11151	11151/M	11151	11151	11151	11151	11151	11151/M
		11151	11151/M	11151	11151/M	11151	11151	11151	11151/M
		11151	11151	11151/M	11151	11151/M	11151	11151	11151
		11151	11151	11151	11151/M	11151	11151/M	11151	11151/M
		11151	11151	11151	11151	11151	11151/M	11151	11151/M
		11151/M							
ptle_p	6679	6681/M	6682/S	6693	6696	6698	6701	6723/M	6724/S
ptle_p	6749	6751/M	6761						
ptle_p	6878	6881/M	6912	6913/M	6913	6915/M	6919	6920/M	6929
		6930/M	6940/S	6989/S	6991/M	7004/S	7007/M	7013	7015/S
		7020/M	7030	7034/M	7037	7042/M	7062/S		
ptle_p	7766	7864/M	7864/S	7864	7864	7864	7864	7864/M	7864/S
ptle_p	7766	7866/M	7866	7866/M	7866	7866/M	7866	7866/M	7866
		7866/M	7866/S	7866/S	7866/M	7866/S	7866/M	7866	7866/S
		7866/M	7866	7866/M	7866	7866/M	7866/S	7898/M	7898
		7898/M	7898	7898/M	7898	7898/M	7898	7898/M	7898/S
		7898/S	7898/M	7898/S	7898/M	7898	7898/S	7898/M	7898
		7898/M	7898	7898/M	7898/S				
ptle_p	7766	7877	7877	7877/S	7877	7877	7877	7877/S	7877
		7877/M	7877/M						
ptle_p	7932	7979/M	7979	7979/M	7979	7979/M	7979	7979/M	7979
		7979/M	7979/S	7979/S	7979/M	7979/S	7979/M	7979	7979/S
		7979/M	7979	7979/M	7979	7979/M	7979/S		
ptle_p	8110	8172/M	8172	8172/M	8172	8172/M	8172	8172/M	8172
		8172/M	8172/S	8172/S	8172/M	8172/S	8172/M	8172	8172/S
		8172/M	8172	8172/M	8172	8172/M	8172/S		
ptle_p	8182	8223	8223	8223/S	8223	8223	8223	8223/S	8223
		8223/M	8223/M	8249	8249	8249/S	8249	8249	8249
		8249/S	8249	8249/M	8249/M				
ptle_p	8182	8227/M	8227	8227/M	8227	8227/M	8227	8227/M	8227
		8227/M	8227/S	8227/S	8227/M	8227/S	8227/M	8227	8227/S
		8227/M	8227	8227/M	8227	8227/M	8227/S		
ptle_p	8359	8362	8363	8365/S	8365	8368	8369	8371/S	8371
		8374/M	8375/M						
ptle_p	8381	8499/M	8499	8499/M	8499	8499/M	8499	8499/M	8499
		8499/M	8499/S	8499/S	8499/M	8499/S	8499/M	8499	8499/S
		8499/M	8499	8499/M	8499	8499/M	8499/S		
ptle_p	8581	8630/M	8630	8630/M	8630	8630/M	8630	8630/M	8630
		8630/M	8630/S	8630/S	8630/M	8630/S	8630/M	8630	8630/S
		8630/M	8630	8630/M	8630	8630/M	8630/S		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ptle_p	8997	9018/M	9020	9023	9024	9026/M	9027/M	9034/M	9036/M
		9038/M	9042/M	9044/M	9047/M				
ptle_p	9091	9134/M	9135/M	9141	9143	9153	9154	9157	9158
		9159/M	9160/M	9163	9169/M	9171/M	9173/M	9180/M	9181/M
ptle_p	9349	9406/M	9406	9406/M	9406	9406/M	9406	9406/M	9406
		9406/M	9406/S	9406/S	9406/M	9406/S	9406/M	9406	9406/S
		9406/M	9406	9406/M	9406	9406/M	9406/S		
ptle_p	9349	9417/M	9417/S	9417	9417	9417	9417	9417/M	9417/S
ptle_p	9446	9479/M	9479	9479/M	9479	9479/M	9479	9479/M	9479
		9479/M	9479/S	9479/S	9479/M	9479/S	9479/M	9479	9479/S
		9479/M	9479	9479/M	9479	9479/M	9479/S		
ptle_p	9491	9528/M	9528/S	9528	9528	9528	9528	9528/M	9528/S
		9562/M	9562/S	9562	9562	9562	9562	9562/M	9562/S
ptle_p	9584	9623/M	9623	9623/M	9623	9623/M	9623	9623/M	9623
		9623/M	9623/S	9623/S	9623/M	9623/S	9623/M	9623	9623/S
		9623/M	9623	9623/M	9623	9623/M	9623/S	9647/M	9647
		9647/M	9647	9647/M	9647	9647/M	9647	9647/M	9647/S
		9647/S	9647/M	9647/S	9647/M	9647	9647/S	9647/M	9647
		9647/M	9647	9647/M	9647/S				
ptle_p	10157	10198/M	10198/S	10198	10198	10198	10198	10198/M	10198/S
ptle_p	10157	10201/M	10201	10201/M	10201	10201/M	10201	10201/M	10201
		10201/M	10201/S	10201/S	10201/M	10201/S	10201/M	10201	10201/S
		10201/M	10201	10201/M	10201	10201/M	10201/S		
ptle_p	10165	10192/M	10197	10199/M	10203/M				
ptle_p	10408	10519/M	10519	10519/M	10519	10519/M	10519	10519/M	10519
		10519/M	10519/S	10519/S	10519/M	10519/S	10519/M	10519	10519/S
		10519/M	10519	10519/M	10519	10519/M	10519/S	10550/M	10550
		10550/M	10550	10550/M	10550	10550/M	10550	10550/M	10550/S
		10550/S	10550/M	10550/S	10550/M	10550	10550/S	10550/M	10550
		10550/M	10550	10550/M	10550/S	10579/M	10579	10579/M	10579
		10579/M	10579	10579/M	10579	10579/M	10579/S	10579/S	10579/M
		10579/S	10579/M	10579	10579/S	10579/M	10579	10579/M	10579
		10579/M	10579/S						
ptle_p	10408	10550/M	10550	10550/S	10550/M	10550	10550	10550/P	10550
		10550/M	10550	10550/M	10550/P	10550	10550/M	10550	
ptle_p	10408	10725/M	10725/S	10725	10725	10725	10725	10725/M	10725/S
ptle_p	10435	10473/M	10480	10527	10528/M	10556	10557/M	10557	10558/M
		10560	10563	10563	10564	10569	10575	10576/M	10581/M
		10582/M	10633/M	10635	10639	10677/M	10678	10682	10781/M
		10784	10787	10790	10793	10794/M	10796	10797	10798/M
		10801	10804/M						
ptle_p	10879	10925/M	10925	10925/M	10925	10925/M	10925	10925/M	10925
		10925/M	10925/S	10925/S	10925/M	10925/S	10925/M	10925	10925/S
		10925/M	10925	10925/M	10925	10925/M	10925/S		
ptle_p	10891	10899/M	10900	10901/S	10902/M	10903	10903	10906/P	10907
		10908/M	10909	10912/M	10913/P	10916	10920/M	10921	
ptle_p	10964	10997/M	10997						
ptle_p	11095	11139/M	11139	11139/M	11139	11139/M	11139	11139/M	11139
		11139/M	11139/S	11139/S	11139/M	11139/S	11139/M	11139	11139/S
		11139/M	11139	11139/M	11139	11139/M	11139/S	11151/M	11151
		11151/M	11151	11151/M	11151	11151/M	11151	11151/M	11151/S
		11151/S	11151/M	11151/S	11151/M	11151	11151/S	11151/M	11151
		11151/M	11151	11151/M	11151/S				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ptle_p	11101	11117/M	11119	11120/M	11120	11121/M	11123	11126	11126
		11127	11132	11138/M					
ptlo	6646	6648/S	6649/S	6650/S	6652	6654	6656		
ptlo	6661	6681/S	6685	6695	6697	6700	6705	6705	6712/S
		6713	6716	6719					
ptlo	6731	6751/S	6755	6760	6764	6764	6771/S	6772	
ptlo	6863	7054	7054	7054/S	7054	7054	7054/S	7054/S	7054
		7054/S	7054	7054/S	7054	7054/S	7054/S	7054	7054/S
		7054							
ptlo	6863	7056	7056	7056/S	7056	7056	7056/S	7056/S	7056
		7056/S	7056	7056/S	7056	7056/S	7056/S	7056/S	7056
		7056/S	7056						
ptlo	6863	7058	7058	7058/S	7058				
ptlo	6866	6881/S	6987	7008	7009	7010	7011	7021	7027
		7029	7031	7035	7036	7038	7041	7043	7054/P
		7056/P	7058/P						
ptlo	7070	7092	7103	7103	7105/S	7126	7146/S	7148/S	7159
		7160/S	7162	7163/S	7166	7168/S	7169/S	7170	7171/S
		7172							
ptlo	7182	7204	7215	7215	7217/S	7241	7255/S	7257/S	7268
		7269/S	7271	7272/S	7274	7275/S	7277/S	7278/S	7279
		7280/S	7281						
ptlo	7291	7314	7317	7317	7319/S				
ptlo	7608	7612/S							
ptlo	7766	7864/S	7864	7864	7864	7864	7864	7864	7864/S
		7864	7864	7864					
ptlo	7766	7866/S	7866	7866	7866	7866	7866	7866	7866
		7866	7866	7866	7866	7866	7866	7866	7866/P
		7866/P	7866/P	7898/S	7898	7898	7898	7898	7898
		7898	7898	7898	7898	7898	7898	7898	7898
		7898	7898/P	7898/P	7898/P				
ptlo	7766	7866	7866	7866/S	7866	7866	7866/S	7866/S	7866
		7866/S	7866	7866/S	7866	7866/S	7866/S	7866	7866/S
		7866	7898	7898	7898/S	7898	7898	7898/S	7898/S
		7898	7898/S	7898	7898/S	7898	7898/S	7898/S	7898
		7898/S	7898						
ptlo	7766	7866	7866	7866/S	7866	7866	7866/S	7866/S	7866
		7866/S	7866	7866/S	7866	7866/S	7866/S	7866/S	7866
		7866/S	7866	7898	7898	7898/S	7898	7898	7898/S
		7898/S	7898	7898/S	7898	7898/S	7898	7898/S	7898/S
		7898/S	7898	7898/S	7898				
ptlo	7766	7866	7866	7866/S	7866	7898	7898	7898/S	7898
ptlo	7932	7979/S	7979	7979	7979	7979	7979	7979	7979
		7979	7979	7979	7979	7979	7979	7979	7979/P
		7979/P	7979/P						
ptlo	7932	7979	7979	7979/S	7979	7979	7979/S	7979/S	7979
		7979/S	7979	7979/S	7979	7979/S	7979/S	7979	7979/S
		7979							
ptlo	7932	7979	7979	7979/S	7979	7979	7979/S	7979/S	7979
		7979/S	7979	7979/S	7979	7979/S	7979/S	7979/S	7979
		7979/S	7979						
ptlo	7932	7979	7979	7979/S	7979				
ptlo	8110	8172/S	8172	8172	8172	8172	8172	8172	8172

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8172	8172	8172	8172	8172	8172	8172	8172/P
		8172/P	8172/P						
ptlo	8110	8172	8172	8172/S	8172	8172	8172/S	8172/S	8172
		8172/S	8172	8172/S	8172	8172/S	8172/S	8172	8172/S
		8172							
ptlo	8110	8172	8172	8172/S	8172	8172	8172/S	8172/S	8172
		8172/S	8172	8172/S	8172	8172/S	8172/S	8172/S	8172
		8172/S	8172						
ptlo	8110	8172	8172	8172/S	8172				
ptlo	8182	8227/S	8227	8227	8227	8227	8227	8227	8227
		8227	8227	8227	8227	8227	8227	8227	8227/P
		8227/P	8227/P						
ptlo	8182	8227	8227	8227/S	8227	8227	8227/S	8227/S	8227
		8227/S	8227	8227/S	8227	8227/S	8227/S	8227	8227/S
		8227							
ptlo	8182	8227	8227	8227/S	8227	8227	8227/S	8227/S	8227
		8227/S	8227	8227/S	8227	8227/S	8227/S	8227/S	8227
		8227/S	8227						
ptlo	8182	8227	8227	8227/S	8227				
ptlo	8276	8288/S	8288	8288	8288	8288	8288	8288/S	8288
		8288	8288/S	8288	8288				
ptlo	8280	8284	8285/S	8285/S	8287/S	8288/P			
ptlo	8299	8321/S	8324	8325	8327	8328	8333	8337/S	8339
		8341	8343/S	8345	8347				
ptlo	8381	8499/S	8499	8499	8499	8499	8499	8499	8499
		8499	8499	8499	8499	8499	8499	8499	8499/P
		8499/P	8499/P						
ptlo	8381	8499	8499	8499/S	8499	8499	8499/S	8499/S	8499
		8499/S	8499	8499/S	8499	8499/S	8499/S	8499	8499/S
		8499							
ptlo	8381	8499	8499	8499/S	8499	8499	8499/S	8499/S	8499
		8499/S	8499	8499/S	8499	8499/S	8499/S	8499/S	8499
		8499/S	8499						
ptlo	8381	8499	8499	8499/S	8499				
ptlo	8513	8544/S	8544/S	8544/S	8544	8544	8544		
ptlo	8581	8630/S	8630	8630	8630	8630	8630	8630	8630
		8630	8630	8630	8630	8630	8630	8630	8630/P
		8630/P	8630/P						
ptlo	8581	8630	8630	8630/S	8630	8630	8630/S	8630/S	8630
		8630/S	8630	8630/S	8630	8630/S	8630/S	8630	8630/S
		8630							
ptlo	8581	8630	8630	8630/S	8630	8630	8630/S	8630/S	8630
		8630/S	8630	8630/S	8630	8630/S	8630/S	8630/S	8630
		8630/S	8630						
ptlo	8581	8630	8630	8630/S	8630				
ptlo	8682	8741/S	8741/S	8741/S	8741	8741	8741		
ptlo	8682	8754/M	8754	8754/S	8754/M	8754/S	8754	8754/S	8754
ptlo	8778	8803/M	8813/S	8834/S	8835/S	8850/S			
ptlo	8894	8896/M	8913/S	8914/S	8927/S	8929/S	8930/S		
ptlo	9217	9255/S							
ptlo	9223	9232/M	9254	9254	9255/P	9259/S	9260/S	9261/S	9262/S
		9264/M	9264/S						
ptlo	9283	9295/M	9297	9297	9298/S	9299/M	9299/S		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ptlo	9349	9406/S	9406	9406	9406	9406	9406	9406	9406
		9406	9406	9406	9406	9406	9406	9406	9406/P
		9406/P	9406/P						
ptlo	9349	9406	9406	9406/S	9406	9406	9406/S	9406/S	9406
		9406/S	9406	9406/S	9406	9406/S	9406/S	9406	9406/S
		9406							
ptlo	9349	9406	9406	9406/S	9406	9406	9406/S	9406/S	9406
		9406/S	9406	9406/S	9406	9406/S	9406/S	9406/S	9406
		9406/S	9406						
ptlo	9349	9406	9406	9406/S	9406				
ptlo	9349	9417/S	9417	9417	9417	9417	9417	9417	9417/S
		9417	9417	9417					
ptlo	9359	9380/M	9384	9384	9385/S	9387/S	9388/S	9389/S	9390/S
		9394/S	9396	9397/M	9398	9398	9399/S	9400/S	9401/S
		9403/S	9406/P	9409/S	9410/M	9410/S	9416/S	9417/P	9418/S
		9420/S	9422/S	9426/S	9429/M	9429/S			
ptlo	9446	9479/S	9479	9479	9479	9479	9479	9479	9479
		9479	9479	9479	9479	9479	9479	9479	9479/P
		9479/P	9479/P						
ptlo	9446	9479	9479	9479/S	9479	9479	9479/S	9479/S	9479
		9479/S	9479	9479/S	9479	9479/S	9479/S	9479	9479/S
		9479							
ptlo	9446	9479	9479	9479/S	9479	9479	9479/S	9479/S	9479
		9479/S	9479	9479/S	9479	9479/S	9479/S	9479/S	9479
		9479/S	9479						
ptlo	9446	9479	9479	9479/S	9479				
ptlo	9455	9464/M	9470	9470	9471/S	9472/S	9473/S	9474/S	9477/S
		9479/P	9482/S	9483/M	9483/S				
ptlo	9491	9528/S	9528	9528	9528	9528	9528	9528	9528/S
		9528	9528	9528	9562/S	9562	9562	9562	9562
		9562	9562	9562/S	9562	9562	9562		
ptlo	9496	9508/M	9514	9514	9515/S	9520/S	9521/S	9522/S	9523/S
		9526/S	9527/S	9528/P	9529/S	9531/S	9533/S	9536/M	9536/S
		9548	9548	9549/S	9554/S	9555/S	9556/S	9557/S	9560/S
		9561/S	9562/P	9563/S	9565/S	9567/S	9570/M	9570/S	
ptlo	9584	9623/S	9623	9623	9623	9623	9623	9623	9623
		9623	9623	9623	9623	9623	9623	9623	9623/P
		9623/P	9623/P	9647/S	9647	9647	9647	9647	9647
		9647	9647	9647	9647	9647	9647	9647	9647
		9647	9647/P	9647/P	9647/P				
ptlo	9584	9623	9623	9623/S	9623	9623	9623/S	9623/S	9623
		9623/S	9623	9623/S	9623	9623/S	9623/S	9623	9623/S
		9623	9647	9647	9647/S	9647	9647	9647/S	9647/S
		9647	9647/S	9647	9647/S	9647	9647/S	9647/S	9647
		9647/S	9647						
ptlo	9584	9623	9623	9623/S	9623	9623	9623/S	9623/S	9623
		9623/S	9623	9623/S	9623	9623/S	9623/S	9623/S	9623
		9623/S	9623	9647	9647	9647/S	9647	9647	9647/S
		9647/S	9647	9647/S	9647	9647/S	9647	9647/S	9647/S
		9647/S	9647	9647/S	9647				
ptlo	9584	9623	9623	9623/S	9623	9647	9647	9647/S	9647
ptlo	9590	9602/M	9611	9611	9612/S	9616/S	9617/S	9618/S	9621/S
		9623/P	9626/S	9628/M	9628/S	9635	9635	9636/S	9640/S

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9641/S	9642/S	9645/S	9647/P	9650/S	9652/M	9652/S	
ptlo	9880	9882/M	9883	9883/S	9883	9883/S	9884/M	9884/S	9886
ptlo	10157	10198/S	10198	10198	10198	10198	10198	10198	10198/S
		10198	10198	10198					
ptlo	10157	10201/S	10201	10201	10201	10201	10201	10201	10201
		10201	10201	10201	10201	10201	10201	10201	10201/P
		10201/P	10201/P						
ptlo	10157	10201	10201	10201/S	10201	10201	10201/S	10201/S	10201
		10201/S	10201	10201/S	10201	10201/S	10201/S	10201	10201/S
		10201							
ptlo	10157	10201	10201	10201/S	10201	10201	10201/S	10201/S	10201
		10201/S	10201	10201/S	10201	10201/S	10201/S	10201/S	10201
		10201/S	10201						
ptlo	10157	10201	10201	10201/S	10201				
ptlo	10164	10191/M	10192/S	10198/P	10201/P				
ptlo	10408	10516/S							
ptlo	10408	10519/S	10519	10519	10519	10519	10519	10519	10519
		10519	10519	10519	10519	10519	10519	10519	10519/P
		10519/P	10519/P	10550/S	10550	10550	10550	10550	10550
		10550	10550	10550	10550	10550	10550	10550	10550
		10550	10550/P	10550/P	10550/P	10579/S	10579	10579	10579
		10579	10579	10579	10579	10579	10579	10579	10579
		10579	10579	10579	10579/P	10579/P	10579/P		
ptlo	10408	10519	10519	10519/S	10519	10519	10519/S	10519/S	10519
		10519/S	10519	10519/S	10519	10519/S	10519/S	10519	10519/S
		10519	10550	10550	10550/S	10550	10550	10550/S	10550/S
		10550	10550/S	10550	10550/S	10550	10550/S	10550/S	10550
		10550/S	10550	10579	10579	10579/S	10579	10579	10579/S
		10579/S	10579	10579/S	10579	10579/S	10579	10579/S	10579/S
		10579	10579/S	10579					
ptlo	10408	10519	10519	10519/S	10519	10519	10519/S	10519/S	10519
		10519/S	10519	10519/S	10519	10519/S	10519/S	10519/S	10519
		10519/S	10519	10550	10550	10550/S	10550	10550	10550/S
		10550/S	10550	10550/S	10550	10550/S	10550	10550/S	10550/S
		10550/S	10550	10550/S	10550	10579	10579	10579/S	10579
		10579	10579/S	10579/S	10579	10579/S	10579	10579/S	10579
		10579/S	10579/S	10579/S	10579	10579/S	10579		
ptlo	10408	10519	10519	10519/S	10519	10550	10550	10550/S	10550
		10579	10579	10579/S	10579				
ptlo	10408	10530/M	10530	10530/S	10530/M	10530/S	10530	10530/S	10530
ptlo	10408	10550/M	10550/S	10550/P					
ptlo	10408	10561/S	10561	10561	10561	10561	10561	10561/S	10561
		10561	10561/S	10561	10561				
ptlo	10408	10725/S	10725	10725	10725	10725	10725	10725	10725/S
		10725	10725	10725					
ptlo	10436	10472/M	10473/S	10524	10561/P	10579/P	10588/M	10598/M	10632/M
		10633/S	10671/M	10674/M	10676	10676	10677/S	10704/M	10704/S
		10723	10725/P	10733	10783	10822			
ptlo	10833	10848/S							
ptlo	10866	10868	10869/S	10870/S	10870/S				
ptlo	10879	10925/S	10925	10925	10925	10925	10925	10925	10925
		10925	10925	10925	10925	10925	10925	10925	10925/P
		10925/P	10925/P						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ptlo	10879	10925	10925	10925/S	10925	10925	10925/S	10925/S	10925
		10925/S	10925	10925/S	10925	10925/S	10925/S	10925	10925/S
		10925							
ptlo	10879	10925	10925	10925/S	10925	10925	10925/S	10925/S	10925
		10925/S	10925	10925/S	10925	10925/S	10925/S	10925/S	10925
		10925/S	10925						
ptlo	10879	10925	10925	10925/S	10925				
ptlo	10890	10898/M	10899/S	10925/P					
ptlo	10964	10997/S	10997	10997	10997	10997	10997/S	10997	
ptlo	10973	10996/M	10997/P	10998/S	11014	11015/S	11016	11018	
ptlo	11068	11086/S	11086/S	11086/S	11086	11086	11086		
ptlo	11095	11124/S	11124	11124	11124	11124	11124	11124/S	11124
		11124	11124/S	11124	11124				
ptlo	11095	11139/S	11139	11139	11139	11139	11139	11139	11139
		11139	11139	11139	11139	11139	11139	11139	11139/P
		11139/P	11139/P	11151/S	11151	11151	11151	11151	11151
		11151	11151	11151	11151	11151	11151	11151	11151
		11151	11151/P	11151/P	11151/P				
ptlo	11095	11139	11139	11139/S	11139	11139	11139/S	11139/S	11139
		11139/S	11139	11139/S	11139	11139/S	11139/S	11139	11139/S
		11139	11151	11151	11151/S	11151	11151	11151/S	11151/S
		11151	11151/S	11151	11151/S	11151	11151/S	11151/S	11151
		11151/S	11151						
ptlo	11095	11139	11139	11139/S	11139	11139	11139/S	11139/S	11139
		11139/S	11139	11139/S	11139	11139/S	11139/S	11139/S	11139
		11139/S	11139	11151	11151	11151/S	11151	11151	11151/S
		11151/S	11151	11151/S	11151	11151/S	11151	11151/S	11151/S
		11151/S	11151	11151/S	11151				
ptlo	11095	11139	11139	11139/S	11139	11151	11151	11151/S	11151
ptlo	11095	11150/S							
ptlo	11102	11116/M	11117/S	11124/P	11139/P				
pva	3161	7595							
pva	3175	7590	7592	8006/M	8006	9007	9168		
pva	4713	8220							
queue	7766	7877/M	7877/M						
queue	8182	8223/M	8223/M	8249/M	8249/M				
queue	8360	8363/M	8369/M						
queue_head	5100	6602	6608/M	6651	6652/M	6684	6685	6689/M	6695
		6696/M	6704	6754	6755	6758/M	6760	6761/M	6763
		6970	6972/M	6981	6982/M	7006	7008/M	7054	7054
		7054/M	7054	7054/M	7056	7056	7056/M	7056	7056/M
		7058	7058	7058/M	7093	7109	7110/M	7159	7160/M
		7205	7221	7222/M	7268	7269/M	7312	7323	7324/M
		7864	7864	7864/M	7864	7864/M	7864	7866	7866/M
		7866	7866/M	7866	7866/M	7866	7866	7866/M	7866
		7866/M	7866	7866	7866/M	7866	7866/M	7866	7866
		7866/M	7898	7898/M	7898	7898/M	7898	7898/M	7898
		7898	7898/M	7898	7898/M	7898	7898	7898/M	7898
		7898/M	7898	7898	7898/M	7979	7979/M	7979	7979/M
		7979	7979/M	7979	7979	7979/M	7979	7979/M	7979
		7979	7979/M	7979	7979/M	7979	7979	7979/M	8172
		8172/M	8172	8172/M	8172	8172/M	8172	8172	8172/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8172	8172/M	8172	8172	8172/M	8172	8172/M	8172
		8172	8172/M	8227	8227/M	8227	8227/M	8227	8227/M
		8227	8227	8227/M	8227	8227/M	8227	8227	8227/M
		8227	8227/M	8227	8227	8227/M	8445	8445/M	8499
		8499/M	8499	8499/M	8499	8499/M	8499	8499	8499/M
		8499	8499/M	8499	8499	8499/M	8499	8499/M	8499
		8499	8499/M	8544	8544/M	8608	8608/M	8630	8630/M
		8630	8630/M	8630	8630/M	8630	8630	8630/M	8630
		8630/M	8630	8630	8630/M	8630	8630/M	8630	8630
		8630/M	8741	8741/M	9406	9406/M	9406	9406/M	9406
		9406/M	9406	9406	9406/M	9406	9406/M	9406	9406
		9406/M	9406	9406/M	9406	9406	9406/M	9417	9417
		9417/M	9417	9417/M	9417	9479	9479/M	9479	9479/M
		9479	9479/M	9479	9479	9479/M	9479	9479/M	9479
		9479	9479/M	9479	9479/M	9479	9479	9479/M	9528
		9528	9528/M	9528	9528/M	9528	9562	9562	9562/M
		9562	9562/M	9562	9623	9623/M	9623	9623/M	9623
		9623/M	9623	9623	9623/M	9623	9623/M	9623	9623
		9623/M	9623	9623/M	9623	9623	9623/M	9647	9647/M
		9647	9647/M	9647	9647/M	9647	9647	9647/M	9647
		9647/M	9647	9647	9647/M	9647	9647/M	9647	9647
		9647/M	10113	10114/M	10198	10198	10198/M	10198	10198/M
		10198	10201	10201/M	10201	10201/M	10201	10201/M	10201
		10201	10201/M	10201	10201/M	10201	10201	10201/M	10201
		10201/M	10201	10201	10201/M	10519	10519/M	10519	10519/M
		10519	10519/M	10519	10519	10519/M	10519	10519/M	10519
		10519	10519/M	10519	10519/M	10519	10519	10519/M	10550
		10550/M	10550	10550/M	10550	10550/M	10550	10550	10550/M
		10550	10550/M	10550	10550	10550/M	10550	10550/M	10550
		10550	10550/M	10579	10579/M	10579	10579/M	10579	10579/M
		10579	10579	10579/M	10579	10579/M	10579	10579	10579/M
		10579	10579/M	10579	10579	10579/M	10674	10725	10725
		10725/M	10725	10725/M	10725	10925	10925/M	10925	10925/M
		10925	10925/M	10925	10925	10925/M	10925	10925/M	10925
		10925	10925/M	10925	10925/M	10925	10925	10925/M	10997
		10997	10997/M	10997	10997/M	10997	11086	11086/M	11139
		11139/M	11139	11139/M	11139	11139/M	11139	11139	11139/M
		11139	11139/M	11139	11139	11139/M	11139	11139/M	11139
		11139	11139/M	11151	11151/M	11151	11151/M	11151	11151/M
		11151	11151	11151/M	11151	11151/M	11151	11151	11151/M
		11151	11151/M	11151	11151	11151/M			
queue_id	2030	11032/M							
queue_link	4938	6630/M	6631/M	7877	7877	7877/S	7877/M	7877	7877
		7877	7877/S	7877/M	7877	7877/M	7877/M	8074	8074
		8086/M	8087/M	8146	8147/M	8148/M	8223	8223	8223/S
		8223/M	8223	8223	8223	8223/S	8223/M	8223	8223/M
		8223/M	8239	8249	8249	8249/S	8249/M	8249	8249
		8249	8249/S	8249/M	8249	8249/M	8249/M	8268	8269/S
		8288	8288/M	8288/M	8288/M	8288/M	8330	8337/M	8341/M
		8343/M	8347/M	8362	8363	8365/S	8365/M	8365	8368
		8369	8371/S	8371/M	8371	8374/M	8375/M	8445/M	8445/M
		8608/M	8608/M	10550	10550/S	10550/M	10550/M	10561	10561/M
		10561/M	10561/M	10561/M	10900	10901/S	10901/M	10902/M	11008/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11009/M	11124	11124/M	11124/M	11124/M	11124/M		
queue_link	8047	8075	8082	8083/M	8084/M	8086/S	8087	8088/M	
queue_link	8111	8135	8142	8142	8143/M	8144/M	8146/M	8147/S	
queue_link	8209	8233/M	8234	8234	8235	8239/M	8239/S		
queue_tail	5103	6654/S	6656/M	6684	6688/M	6713	6714/M	6754	6759/M
		6772	6773/M	6973/M	7009/M	7031/M	7038/M	7041/S	7043/M
		7054/M	7054/M	7056/M	7056/M	7058/M	7134/M	7172/M	7244/M
		7281/M	7347/M	7864	7864/M	7864	7864/M	7866/M	7866/M
		7866/M	7866/M	7866/S	7866/M	7866/M	7866/M	7866/M	7866/M
		7866/M	7898/M	7898/M	7898/M	7898/M	7898/S	7898/M	7898/M
		7898/M	7898/M	7898/M	7898/M	7979/M	7979/M	7979/M	7979/M
		7979/S	7979/M	7979/M	7979/M	7979/M	7979/M	7979/M	8172/M
		8172/M	8172/M	8172/M	8172/S	8172/M	8172/M	8172/M	8172/M
		8172/M	8172/M	8227/M	8227/M	8227/M	8227/M	8227/S	8227/M
		8227/M	8227/M	8227/M	8227/M	8227/M	8499/M	8499/M	8499/M
		8499/M	8499/S	8499/M	8499/M	8499/M	8499/M	8499/M	8499/M
		8544/S	8544/M	8630/M	8630/M	8630/M	8630/M	8630/S	8630/M
		8630/M	8630/M	8630/M	8630/M	8630/M	8741/S	8741/M	9406/M
		9406/M	9406/M	9406/M	9406/S	9406/M	9406/M	9406/M	9406/M
		9406/M	9406/M	9417	9417/M	9417	9417/M	9479/M	9479/M
		9479/M	9479/M	9479/S	9479/M	9479/M	9479/M	9479/M	9479/M
		9479/M	9528	9528/M	9528	9528/M	9562	9562/M	9562
		9562/M	9623/M	9623/M	9623/M	9623/M	9623/S	9623/M	9623/M
		9623/M	9623/M	9623/M	9623/M	9647/M	9647/M	9647/M	9647/M
		9647/S	9647/M	9647/M	9647/M	9647/M	9647/M	9647/M	10116/S
		10118/M	10198	10198/M	10198	10198/M	10201/M	10201/M	10201/M
		10201/M	10201/S	10201/M	10201/M	10201/M	10201/M	10201/M	10201/M
		10519/M	10519/M	10519/M	10519/M	10519/S	10519/M	10519/M	10519/M
		10519/M	10519/M	10519/M	10550/M	10550/M	10550/M	10550/M	10550/S
		10550/M	10550/M	10550/M	10550/M	10550/M	10550/M	10579/M	10579/M
		10579/M	10579/M	10579/S	10579/M	10579/M	10579/M	10579/M	10579/M
		10579/M	10725	10725/M	10725	10725/M	10925/M	10925/M	10925/M
		10925/M	10925/S	10925/M	10925/M	10925/M	10925/M	10925/M	10925/M
		10997	10997/M	10997	10997/M	11086/S	11086/M	11139/M	11139/M
		11139/M	11139/M	11139/S	11139/M	11139/M	11139/M	11139/M	11139/M
		11139/M	11151/M	11151/M	11151/M	11151/M	11151/S	11151/M	11151/M
		11151/M	11151/M	11151/M	11151/M				
r	6412	10669/M	10720/M						
r	10431	10668/M	10669	10719/M	10720				
rb	8382	8426	8427/P	8428/P	8429	8433/M	8437	8438/S	8443
		8445/P	8445/P	8446	8450/M	8454	8459	8461	8470
		8475	8493/P	8499/P					
rb	8582	8605	8608/P	8608/P	8609	8615/S	8616/S	8630/P	8635/P
		8646							
rb	8683	8702/M	8703	8704/P	8717/P	8717/P	8717/P	8718	
rb	8967	9032	9033	9035	9041	9047			
rb	9058	9101	9102/P	9102/P	9103	9112	9113	9115	9121
		9173	9187	9194/P	9196/S				
rb	9901	9920/M	9921/M	9929/M					
rb	9937	9950/M	9952	9954	9961	9962	9963	9964	9969/P
		9974	9976/P	9980	9991	9993	9997/P	10000	10002
		10005/P	10017	10020/P	10033/P	10040	10047/P	10057/S	10059

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10060	10064/P	10066	10068	10071/P	10088/P	10091/P	10094/P
		10097/P	10105	10106/P	10108/M	10110/M	10111		
rb	10965	10978/M	10979	10983	10984/S	10986/M	10987/M	10993/S	10993/S
		10995	10996	11001	11002	11003	11006	11026/P	11035/P
		11039							
readied_ijle_p	9092	9196/P	9197	9201	9202	9203	9205	9206	
ready	5536	7904/M	9013/M	10451					
ready_condition	7769	7921							
ready_task_count	1646	7052	7866	7889/M	7889	7894/M	7894	7898	7970/M
		7970	7975/M	7975	7979	8163/M	8163	8168/M	8168
		8172	8224/M	8224	8227	8250/M	8250	8468/M	8499
		8630	8642/M	8642	8729/M	8729	9249	9406	9479
		9623	9647	10201	10519	10550	10550/M	10550	10559/M
		10559	10578	10579	10925	10927/M	10927	11122/M	11122
		11139	11151						
ready_task_count	9360	9381/M	9419/M	9419					
ready_task_count	9453	9468/M	9475/M	9475					
ready_task_count	9498	9509/M	9530/M	9530	9564/M	9564			
ready_task_count	9591	9606/M	9619/M	9619	9643/M	9643			
ready_tasks	4817	6692/M	6692	6694/M	6694	6694	6694	6694/M	6694
		6976/M	6976	6979/M	6979	6979	6979	6979/M	6979
		7012/M	7012	7014/M	7014	7014	7014	7014/M	7014
		7864/M	7864	7864/M	7864	7864	7864	7864/M	7864
		7866/M	7866	7866/M	7866	7866	7866	7866/M	7866
		7866/M	7866	7866/M	7866	7866	7866	7866/M	7866
		7898/M	7898	7898/M	7898	7898	7898	7898/M	7898
		7898/M	7898	7898/M	7898	7898	7898	7898/M	7898
		7979/M	7979	7979/M	7979	7979	7979	7979/M	7979
		7979/M	7979	7979/M	7979	7979	7979	7979/M	7979
		8172/M	8172	8172/M	8172	8172	8172	8172/M	8172
		8172/M	8172	8172/M	8172	8172	8172	8172/M	8172
		8227/M	8227	8227/M	8227	8227	8227	8227/M	8227
		8227/M	8227	8227/M	8227	8227	8227	8227/M	8227
		8499/M	8499	8499/M	8499	8499	8499	8499/M	8499
		8499/M	8499	8499/M	8499	8499	8499	8499/M	8499
		8630/M	8630	8630/M	8630	8630	8630	8630/M	8630
		8630/M	8630	8630/M	8630	8630	8630	8630/M	8630
		8739/M	8739	8739	8739	8739/M	8739	8739/M	8739
		8739	8739/M	8739	8739/M	8739	8739	8739	8739/M
		8739	8739/M	8739	8739	8739/M	8739	8739/M	8739
		8739	8739	8739/M	8739	8739/M	8739	8739	8739/M
		8739	9406/M	9406	9406/M	9406	9406	9406	9406/M
		9406	9406/M	9406	9406/M	9406	9406	9406	9406/M
		9406	9417/M	9417	9417/M	9417	9417	9417	9417/M
		9417	9479/M	9479	9479/M	9479	9479	9479	9479/M
		9479	9479/M	9479	9479/M	9479	9479	9479	9479/M
		9479	9528/M	9528	9528/M	9528	9528	9528	9528/M
		9528	9562/M	9562	9562/M	9562	9562	9562	9562/M
		9562	9623/M	9623	9623/M	9623	9623	9623	9623/M
		9623	9623/M	9623	9623/M	9623	9623	9623	9623/M
		9623	9647/M	9647	9647/M	9647	9647	9647	9647/M
		9647	9647/M	9647	9647/M	9647	9647	9647	9647/M
		9647	9694/M	9694	9694	9694	9694/M	9694	9694/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9694	9694	9694/M	9694	9694/M	9694	9694	9694
		9694/M	9694	9694/M	9694	9694	9694/M	9694	9694/M
		9694	9694	9694	9694/M	9694	9694/M	9694	9694
		9694/M	9694	9766/M	9766	9766	9766	9766/M	9766
		9768/M	9768	9768	9768/M	9768	9781/M	9781	9781
		9781	9781/M	9781	9782/M	9782	9782	9782/M	9782
		9806/M	9806	9806	9806	9806/M	9806	9807/M	9807
		9807	9807/M	9807	9834/M	9837	9838	9839/M	9839
		9858/M	9858	9861	9862	9863/M	9863	10198/M	10198
		10198/M	10198	10198	10198	10198/M	10198	10201/M	10201
		10201/M	10201	10201	10201	10201/M	10201	10201/M	10201
		10201/M	10201	10201	10201	10201/M	10201	10466/M	10466
		10466	10466	10466/M	10466	10466/M	10466	10466	10466/M
		10466	10466/M	10466	10466	10466	10466/M	10466	10466/M
		10466	10466	10466/M	10466	10466/M	10466	10466	10466
		10466/M	10466	10466/M	10466	10466	10466/M	10466	10519/M
		10519	10519/M	10519	10519	10519	10519/M	10519	10519/M
		10519	10519/M	10519	10519	10519	10519/M	10519	10550/M
		10550	10550/M	10550	10550	10550	10550/M	10550	10550/M
		10550	10550/M	10550	10550	10550	10550/M	10550	10579/M
		10579	10579/M	10579	10579	10579	10579/M	10579	10579/M
		10579	10579/M	10579	10579	10579	10579/M	10579	10658/M
		10658	10661	10662	10663/M	10663	10712/M	10712	10725/M
		10725	10725/M	10725	10725	10725	10725/M	10725	10774/M
		10774	10774	10774	10774/M	10774	10774/M	10774	10774
		10774/M	10774	10774/M	10774	10774	10774	10774/M	10774
		10774/M	10774	10774	10774/M	10774	10774/M	10774	10774
		10774	10774/M	10774	10774/M	10774	10774	10774/M	10774
		10925/M	10925	10925/M	10925	10925	10925	10925/M	10925
		10925/M	10925	10925/M	10925	10925	10925	10925/M	10925
		11139/M	11139	11139/M	11139	11139	11139	11139/M	11139
		11139/M	11139	11139/M	11139	11139	11139	11139/M	11139
		11151/M	11151	11151/M	11151	11151	11151	11151/M	11151
		11151/M	11151	11151/M	11151	11151	11151	11151/M	11151
readying_task_priority	4943	6633/M	6912	6913	7838	7839/M	7866	7866	7898
		7898	7979	7979	8172	8172	8227	8227	8445/M
		8499	8499	8608/M	8630	8630	8835/M	8842	8844
		8914/M	8919	8921	9189	9192	9406	9406	9479
		9479	9623	9623	9647	9647	10201	10201	10519
		10519	10550	10550	10579	10579	10925	10925	11139
		11139	11151	11151					
readying_task_priority	7768	7836	7838	7839	7842	7843			
readying_task_priority	9093	9190/M	9192/M	9194/P					
register_id	4883	9954							
relative_insert_major_dct	7180	7056	7286	7866	7898	7979	8172	8227	8499
		8630	9406	9479	9623	9647	10201	10519	10550
		10579	10925	11139	11151				
relative_insert_minor_dct	7068	7054	7177	7866	7898	7979	8172	8227	8499
		8630	9406	9479	9623	9647	10201	10519	10550
		10579	10925	11139	11151				
relative_insert_tail_dct	7289	7058	7349	7866	7898	7979	8172	8227	8499
		8630	9406	9479	9623	9647	10201	10519	10550
		10579	10925	11139	11151				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
relative_priority	6863	7054	7054						
relative_priority	6863	7056	7056						
relative_priority	6863	7058							
relative_priority	7069	7108	7151						
relative_priority	7181	7220	7261						
relative_priority	7290	7322							
relative_priority	7766	7866	7866	7898	7898				
relative_priority	7766	7866	7866	7898	7898				
relative_priority	7766	7866	7898						
relative_priority	7932	7979	7979						
relative_priority	7932	7979	7979						
relative_priority	7932	7979							
relative_priority	8110	8172	8172						
relative_priority	8110	8172	8172						
relative_priority	8110	8172							
relative_priority	8182	8227	8227						
relative_priority	8182	8227	8227						
relative_priority	8182	8227							
relative_priority	8381	8499	8499						
relative_priority	8381	8499	8499						
relative_priority	8381	8499							
relative_priority	8581	8630	8630						
relative_priority	8581	8630	8630						
relative_priority	8581	8630							
relative_priority	9349	9406	9406						
relative_priority	9349	9406	9406						
relative_priority	9349	9406							
relative_priority	9446	9479	9479						
relative_priority	9446	9479	9479						
relative_priority	9446	9479							
relative_priority	9584	9623	9623	9647	9647				
relative_priority	9584	9623	9623	9647	9647				
relative_priority	9584	9623	9647						
relative_priority	10157	10201	10201						
relative_priority	10157	10201	10201						
relative_priority	10157	10201							
relative_priority	10408	10519	10519	10550	10550	10579	10579		
relative_priority	10408	10519	10519	10550	10550	10579	10579		
relative_priority	10408	10519	10550	10579					
relative_priority	10879	10925	10925						
relative_priority	10879	10925	10925						
relative_priority	10879	10925							
relative_priority	11095	11139	11139	11151	11151				
relative_priority	11095	11139	11139	11151	11151				
relative_priority	11095	11139	11151						
relative_priority_enabled	1308	7051	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10519	10550	10579
		10925	11139	11151					
relative_task_priority	2544	7054/P	7054	7054	7056/P	7056	7056	7058/P	7058
		7108	7151	7220	7261	7322	7866/P	7866	7866
		7866/P	7866	7866	7866/P	7866	7898/P	7898	7898
		7898/P	7898	7898	7898/P	7898	7979/P	7979	7979
		7979/P	7979	7979	7979/P	7979	8172/P	8172	8172

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8172/P	8172	8172	8172/P	8172	8227/P	8227	8227
		8227/P	8227	8227	8227/P	8227	8499/P	8499	8499
		8499/P	8499	8499	8499/P	8499	8630/P	8630	8630
		8630/P	8630	8630	8630/P	8630	9406/P	9406	9406
		9406/P	9406	9406	9406/P	9406	9479/P	9479	9479
		9479/P	9479	9479	9479/P	9479	9623/P	9623	9623
		9623/P	9623	9623	9623/P	9623	9647/P	9647	9647
		9647/P	9647	9647	9647/P	9647	10201/P	10201	10201
		10201/P	10201	10201	10201/P	10201	10519/P	10519	10519
		10519/P	10519	10519	10519/P	10519	10550/P	10550	10550
		10550/P	10550	10550	10550/P	10550	10579/P	10579	10579
		10579/P	10579	10579	10579/P	10579	10925/P	10925	10925
		10925/P	10925	10925	10925/P	10925	11139/P	11139	11139
		11139/P	11139	11139	11139/P	11139	11151/P	11151	11151
		11151/P	11151	11151	11151/P	11151			
remove_ijl	7375	7413	8742						
remove_task_from_free_queue	6730	6778	10997						
remove_task_from_seg_lock_q	8182	7914	8256						
req	83	7902	7903	9005	9006	10452/P			
reqcode	4876	7813/M							
request_origin	4888	7815/M	9980	9993	10040				
requested_wait_time	4844	9032	9047						
requested_wait_time	4857	9112	9173						
residence	1403	5246	5293	8222	8232				
residence	5244	5246/M	5250	5251					
residence	5286	5293/M	5293	5293					
residence	8182	8222/M	8222	8222	8232/M	8232	8232		
rindex	43	54	55						
ring	1997	7590	7592	9007	9168				
ring	7564	7588/S	7590	7592	7595/S				
s	8405	8473/M							
save_ptlo	6678	6706/M	6712/S	6714	6717	6720			
save_ptlo	6747	6765/M	6771/S	6773					
save_ptlo	6863	7054/M	7054/S	7054/M	7054/M	7054/S			
save_ptlo	6863	7056/M	7056/S	7056/M	7056/M	7056/S			
save_ptlo	6863	7058/M	7058	7058/S	7058/M	7058/M			
save_ptlo	7083	7094/M	7112/S	7118/M	7121/M	7168/S			
save_ptlo	7195	7206/M	7224/S	7233/M	7236/M	7277/S			
save_ptlo	7303	7313/M	7332	7333/S	7339/M	7342/M			
save_ptlo	7766	7864/M	7864/S	7864	7864	7864			
save_ptlo	7766	7866/M	7866/S	7866/M	7866/M	7866/S	7898/M	7898/S	7898/M
		7898/M	7898/S						
save_ptlo	7766	7866/M	7866/S	7866/M	7866/M	7866/S	7898/M	7898/S	7898/M
		7898/M	7898/S						
save_ptlo	7766	7866/M	7866	7866/S	7866/M	7866/M	7898/M	7898	7898/S
		7898/M	7898/M						
save_ptlo	7932	7979/M	7979/S	7979/M	7979/M	7979/S			
save_ptlo	7932	7979/M	7979/S	7979/M	7979/M	7979/S			
save_ptlo	7932	7979/M	7979	7979/S	7979/M	7979/M			
save_ptlo	8110	8172/M	8172/S	8172/M	8172/M	8172/S			
save_ptlo	8110	8172/M	8172/S	8172/M	8172/M	8172/S			
save_ptlo	8110	8172/M	8172	8172/S	8172/M	8172/M			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
save_ptlo	8182	8227/M	8227/S	8227/M	8227/M	8227/S			
save_ptlo	8182	8227/M	8227/S	8227/M	8227/M	8227/S			
save_ptlo	8182	8227/M	8227	8227/S	8227/M	8227/M			
save_ptlo	8381	8499/M	8499/S	8499/M	8499/M	8499/S			
save_ptlo	8381	8499/M	8499/S	8499/M	8499/M	8499/S			
save_ptlo	8381	8499/M	8499	8499/S	8499/M	8499/M			
save_ptlo	8581	8630/M	8630/S	8630/M	8630/M	8630/S			
save_ptlo	8581	8630/M	8630/S	8630/M	8630/M	8630/S			
save_ptlo	8581	8630/M	8630	8630/S	8630/M	8630/M			
save_ptlo	9349	9406/M	9406/S	9406/M	9406/M	9406/S			
save_ptlo	9349	9406/M	9406/S	9406/M	9406/M	9406/S			
save_ptlo	9349	9406/M	9406	9406/S	9406/M	9406/M			
save_ptlo	9349	9417/M	9417/S	9417	9417	9417			
save_ptlo	9446	9479/M	9479/S	9479/M	9479/M	9479/S			
save_ptlo	9446	9479/M	9479/S	9479/M	9479/M	9479/S			
save_ptlo	9446	9479/M	9479	9479/S	9479/M	9479/M			
save_ptlo	9491	9528/M	9528/S	9528	9528	9528	9562/M	9562/S	9562
		9562	9562						
save_ptlo	9584	9623/M	9623/S	9623/M	9623/M	9623/S	9647/M	9647/S	9647/M
		9647/M	9647/S						
save_ptlo	9584	9623/M	9623/S	9623/M	9623/M	9623/S	9647/M	9647/S	9647/M
		9647/M	9647/S						
save_ptlo	9584	9623/M	9623	9623/S	9623/M	9623/M	9647/M	9647	9647/S
		9647/M	9647/M						
save_ptlo	10157	10198/M	10198/S	10198	10198	10198			
save_ptlo	10157	10201/M	10201/S	10201/M	10201/M	10201/S			
save_ptlo	10157	10201/M	10201/S	10201/M	10201/M	10201/S			
save_ptlo	10157	10201/M	10201	10201/S	10201/M	10201/M			
save_ptlo	10408	10519/M	10519/S	10519/M	10519/M	10519/S	10550/M	10550/S	10550/M
		10550/M	10550/S	10579/M	10579/S	10579/M	10579/M	10579/S	
save_ptlo	10408	10519/M	10519/S	10519/M	10519/M	10519/S	10550/M	10550/S	10550/M
		10550/M	10550/S	10579/M	10579/S	10579/M	10579/M	10579/S	
save_ptlo	10408	10519/M	10519	10519/S	10519/M	10519/M	10550/M	10550	10550/S
		10550/M	10550/M	10579/M	10579	10579/S	10579/M	10579/M	
save_ptlo	10408	10725/M	10725/S	10725	10725	10725			
save_ptlo	10879	10925/M	10925/S	10925/M	10925/M	10925/S			
save_ptlo	10879	10925/M	10925/S	10925/M	10925/M	10925/S			
save_ptlo	10879	10925/M	10925	10925/S	10925/M	10925/M			
save_ptlo	10964	10997/M	10997/S	10997					
save_ptlo	11095	11139/M	11139/S	11139/M	11139/M	11139/S	11151/M	11151/S	11151/M
		11151/M	11151/S						
save_ptlo	11095	11139/M	11139/S	11139/M	11139/M	11139/S	11151/M	11151/S	11151/M
		11151/M	11151/S						
save_ptlo	11095	11139/M	11139	11139/S	11139/M	11139/M	11151/M	11151	11151/S
		11151/M	11151/M						
scan_ptlo	6677	6704/M	6705	6705	6706	6707/M	6707/S	6708	
scan_ptlo	6748	6763/M	6764	6764	6764	6764	6765	6766/M	6766/S
		6768							
scan_ptlo	7766	7864/M	7864	7864	7864/M	7864/S	7864	7864	
scan_ptlo	9349	9417/M	9417	9417	9417/M	9417/S	9417	9417	
scan_ptlo	9491	9528/M	9528	9528	9528/M	9528/S	9528	9528	9562/M
		9562	9562	9562/M	9562/S	9562	9562		
scan_ptlo	10157	10198/M	10198	10198	10198/M	10198/S	10198	10198	

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
scan_ptlo	10408	10725/M	10725	10725	10725/M	10725/S	10725	10725	
scan_ptlo	10964	10997/M	10997	10997	10997	10997/M	10997/S	10997	10997
		10997							
scheduler_initiated_jobs	6251	8504/M	8506	8550/M	8551	11046/M	11047		
scheduling_dispatching_priority	1279	7842	7843/M	8848/M	8925/M	10009	10010/M	10028/M	10322
		10343/M							
scheduling_priority	2583	9201	9205						
sdt_offset	2538	8425							
sdt_p	8401	8425/M	8427/P						
sdtx_offset	2539	5571	8221						
sdtx_p	8211	8221/M	8222/P						
search	4827	7658/M	7683						
search	7619	7657	7658	7659	7670				
search_loop	7401	7401	7404	7408					
search_loop	8229	8229	8237	8243					
search_loop	8682	8742	8742	8742					
search_ptlo	6863	7054/M	7054	7054/S	7054/S	7054/S	7054	7054/S	7054/S
		7054/S	7054	7054	7054	7054	7054/M	7054	7054/M
		7054	7054/S	7054/S	7054	7054/M	7054/S	7054	
search_ptlo	6863	7056/M	7056	7056/S	7056/S	7056/S	7056	7056/S	7056/S
		7056	7056/S	7056/S	7056	7056	7056	7056	7056/M
		7056	7056/M	7056	7056/S	7056/S	7056	7056/M	7056/S
		7056							
search_ptlo	6863	7058/M	7058	7058/S	7058/S	7058/S	7058	7058/S	7058
		7058/S	7058	7058/S	7058/S	7058/S	7058	7058	7058
		7058	7058/M	7058					
search_ptlo	7084	7093/M	7103	7103	7104/S	7105/S	7107/S	7109	7110/S
		7112/S	7114/S	7115	7116	7118	7121	7123/M	7146/M
		7147	7147	7148/S	7150/S	7152	7155/M	7155/S	
search_ptlo	7196	7205/M	7215	7215	7216/S	7217/S	7219/S	7221	7222/S
		7224/S	7226	7227/S	7229/S	7230	7231	7233	7236
		7238/M	7255/M	7256	7256	7257/S	7259/S	7262	7265/M
		7265/S							
search_ptlo	7304	7312/M	7317	7317	7318/S	7319/S	7321/S	7323	7324/S
		7326	7327/S	7329	7330/S	7333/S	7335/S	7336	7337
		7339	7342	7344/M					
search_ptlo	7766	7866/M	7866	7866/S	7866/S	7866/S	7866	7866/S	7866/S
		7866/S	7866	7866	7866	7866	7866/M	7866	7866/M
		7866	7866/S	7866/S	7866	7866/M	7866/S	7866	7898/M
		7898	7898/S	7898/S	7898/S	7898	7898/S	7898/S	7898/S
		7898	7898	7898	7898	7898/M	7898	7898/M	7898
		7898/S	7898/S	7898	7898/M	7898/S	7898		
search_ptlo	7766	7866/M	7866	7866/S	7866/S	7866/S	7866	7866/S	7866/S
		7866	7866/S	7866/S	7866	7866	7866	7866	7866/M
		7866	7866/M	7866	7866/S	7866/S	7866	7866/M	7866/S
		7866	7898/M	7898	7898/S	7898/S	7898/S	7898	7898/S
		7898/S	7898	7898/S	7898/S	7898	7898	7898	7898
		7898/M	7898	7898/M	7898	7898/S	7898/S	7898	7898/M
		7898/S	7898						
search_ptlo	7766	7866/M	7866	7866/S	7866/S	7866/S	7866	7866/S	7866
		7866/S	7866	7866/S	7866/S	7866/S	7866	7866	7866
		7866	7866/M	7866	7898/M	7898	7898/S	7898/S	7898/S
		7898	7898/S	7898	7898/S	7898	7898/S	7898/S	7898/S

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7898	7898	7898	7898	7898/M	7898		
search_ptlo	7932	7979/M	7979	7979/S	7979/S	7979/S	7979	7979/S	7979/S
		7979/S	7979	7979	7979	7979	7979/M	7979	7979/M
		7979	7979/S	7979/S	7979	7979/M	7979/S	7979	
search_ptlo	7932	7979/M	7979	7979/S	7979/S	7979/S	7979	7979/S	7979/S
		7979	7979/S	7979/S	7979	7979	7979	7979	7979/M
		7979	7979/M	7979	7979/S	7979/S	7979	7979/M	7979/S
		7979							
search_ptlo	7932	7979/M	7979	7979/S	7979/S	7979/S	7979	7979/S	7979
		7979/S	7979	7979/S	7979/S	7979/S	7979	7979	7979
		7979	7979/M	7979					
search_ptlo	8110	8172/M	8172	8172/S	8172/S	8172/S	8172	8172/S	8172/S
		8172/S	8172	8172	8172	8172	8172/M	8172	8172/M
		8172	8172/S	8172/S	8172	8172/M	8172/S	8172	
search_ptlo	8110	8172/M	8172	8172/S	8172/S	8172/S	8172	8172/S	8172/S
		8172	8172/S	8172/S	8172	8172	8172	8172	8172/M
		8172	8172/M	8172	8172/S	8172/S	8172	8172/M	8172/S
		8172							
search_ptlo	8110	8172/M	8172	8172/S	8172/S	8172/S	8172	8172/S	8172
		8172/S	8172	8172/S	8172/S	8172/S	8172	8172	8172
		8172	8172/M	8172					
search_ptlo	8182	8227/M	8227	8227/S	8227/S	8227/S	8227	8227/S	8227/S
		8227/S	8227	8227	8227	8227	8227/M	8227	8227/M
		8227	8227/S	8227/S	8227	8227/M	8227/S	8227	
search_ptlo	8182	8227/M	8227	8227/S	8227/S	8227/S	8227	8227/S	8227/S
		8227	8227/S	8227/S	8227	8227	8227	8227	8227/M
		8227	8227/M	8227	8227/S	8227/S	8227	8227/M	8227/S
		8227							
search_ptlo	8182	8227/M	8227	8227/S	8227/S	8227/S	8227	8227/S	8227
		8227/S	8227	8227/S	8227/S	8227/S	8227	8227	8227
		8227	8227/M	8227					
search_ptlo	8381	8499/M	8499	8499/S	8499/S	8499/S	8499	8499/S	8499/S
		8499/S	8499	8499	8499	8499	8499/M	8499	8499/M
		8499	8499/S	8499/S	8499	8499/M	8499/S	8499	
search_ptlo	8381	8499/M	8499	8499/S	8499/S	8499/S	8499	8499/S	8499/S
		8499	8499/S	8499/S	8499	8499	8499	8499	8499/M
		8499	8499/M	8499	8499/S	8499/S	8499	8499/M	8499/S
		8499							
search_ptlo	8381	8499/M	8499	8499/S	8499/S	8499/S	8499	8499/S	8499
		8499/S	8499	8499/S	8499/S	8499/S	8499	8499	8499
		8499	8499/M	8499					
search_ptlo	8581	8630/M	8630	8630/S	8630/S	8630/S	8630	8630/S	8630/S
		8630/S	8630	8630	8630	8630	8630/M	8630	8630/M
		8630	8630/S	8630/S	8630	8630/M	8630/S	8630	
search_ptlo	8581	8630/M	8630	8630/S	8630/S	8630/S	8630	8630/S	8630/S
		8630	8630/S	8630/S	8630	8630	8630	8630	8630/M
		8630	8630/M	8630	8630/S	8630/S	8630	8630/M	8630/S
		8630							
search_ptlo	8581	8630/M	8630	8630/S	8630/S	8630/S	8630	8630/S	8630
		8630/S	8630	8630/S	8630/S	8630/S	8630	8630	8630
		8630	8630/M	8630					
search_ptlo	9349	9406/M	9406	9406/S	9406/S	9406/S	9406	9406/S	9406/S
		9406/S	9406	9406	9406	9406	9406/M	9406	9406/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9406	9406/S	9406/S	9406	9406/M	9406/S	9406	
search_ptlo	9349	9406/M	9406	9406/S	9406/S	9406/S	9406	9406/S	9406/S
		9406	9406/S	9406/S	9406	9406	9406	9406	9406/M
		9406	9406/M	9406	9406/S	9406/S	9406	9406/M	9406/S
		9406							
search_ptlo	9349	9406/M	9406	9406/S	9406/S	9406/S	9406	9406/S	9406
		9406/S	9406	9406/S	9406/S	9406/S	9406	9406	9406
		9406	9406/M	9406					
search_ptlo	9446	9479/M	9479	9479/S	9479/S	9479/S	9479	9479/S	9479/S
		9479/S	9479	9479	9479	9479	9479/M	9479	9479/M
		9479	9479/S	9479/S	9479	9479/M	9479/S	9479	
search_ptlo	9446	9479/M	9479	9479/S	9479/S	9479/S	9479	9479/S	9479/S
		9479	9479/S	9479/S	9479	9479	9479	9479	9479/M
		9479	9479/M	9479	9479/S	9479/S	9479	9479/M	9479/S
		9479							
search_ptlo	9446	9479/M	9479	9479/S	9479/S	9479/S	9479	9479/S	9479
		9479/S	9479	9479/S	9479/S	9479/S	9479	9479	9479
		9479	9479/M	9479					
search_ptlo	9584	9623/M	9623	9623/S	9623/S	9623/S	9623	9623/S	9623/S
		9623/S	9623	9623	9623	9623	9623/M	9623	9623/M
		9623	9623/S	9623/S	9623	9623/M	9623/S	9623	9647/M
		9647	9647/S	9647/S	9647/S	9647	9647/S	9647/S	9647/S
		9647	9647	9647	9647	9647/M	9647	9647/M	9647
		9647/S	9647/S	9647	9647/M	9647/S	9647		
search_ptlo	9584	9623/M	9623	9623/S	9623/S	9623/S	9623	9623/S	9623/S
		9623	9623/S	9623/S	9623	9623	9623	9623	9623/M
		9623	9623/M	9623	9623/S	9623/S	9623	9623/M	9623/S
		9623	9647/M	9647	9647/S	9647/S	9647/S	9647	9647/S
		9647/S	9647	9647/S	9647/S	9647	9647	9647	9647
		9647/M	9647	9647/M	9647	9647/S	9647/S	9647	9647/M
		9647/S	9647						
search_ptlo	9584	9623/M	9623	9623/S	9623/S	9623/S	9623	9623/S	9623
		9623/S	9623	9623/S	9623/S	9623/S	9623	9623	9623
		9623	9623/M	9623	9647/M	9647	9647/S	9647/S	9647/S
		9647	9647/S	9647	9647/S	9647	9647/S	9647/S	9647/S
		9647	9647	9647	9647	9647/M	9647		
search_ptlo	10157	10201/M	10201	10201/S	10201/S	10201/S	10201	10201/S	10201/S
		10201/S	10201	10201	10201	10201	10201/M	10201	10201/M
		10201	10201/S	10201/S	10201	10201/M	10201/S	10201	
search_ptlo	10157	10201/M	10201	10201/S	10201/S	10201/S	10201	10201/S	10201/S
		10201	10201/S	10201/S	10201	10201	10201	10201	10201/M
		10201	10201/M	10201	10201/S	10201/S	10201	10201/M	10201/S
		10201							
search_ptlo	10157	10201/M	10201	10201/S	10201/S	10201/S	10201	10201/S	10201
		10201/S	10201	10201/S	10201/S	10201/S	10201	10201	10201
		10201	10201/M	10201					
search_ptlo	10408	10519/M	10519	10519/S	10519/S	10519/S	10519	10519/S	10519/S
		10519/S	10519	10519	10519	10519	10519/M	10519	10519/M
		10519	10519/S	10519/S	10519	10519/M	10519/S	10519	10550/M
		10550	10550/S	10550/S	10550/S	10550	10550/S	10550/S	10550/S
		10550	10550	10550	10550	10550/M	10550	10550/M	10550
		10550/S	10550/S	10550	10550/M	10550/S	10550	10579/M	10579
		10579/S	10579/S	10579/S	10579	10579/S	10579/S	10579/S	10579

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10579	10579	10579	10579/M	10579	10579/M	10579	10579/S
		10579/S	10579	10579/M	10579/S	10579			
search_ptlo	10408	10519/M	10519	10519/S	10519/S	10519/S	10519	10519/S	10519/S
		10519	10519/S	10519/S	10519	10519	10519	10519	10519/M
		10519	10519/M	10519	10519/S	10519/S	10519	10519/M	10519/S
		10519	10550/M	10550	10550/S	10550/S	10550/S	10550	10550/S
		10550/S	10550	10550/S	10550/S	10550	10550	10550	10550
		10550/M	10550	10550/M	10550	10550/S	10550/S	10550	10550/M
		10550/S	10550	10579/M	10579	10579/S	10579/S	10579/S	10579
		10579/S	10579/S	10579	10579/S	10579/S	10579	10579	10579
		10579	10579/M	10579	10579/M	10579	10579/S	10579/S	10579
		10579/M	10579/S	10579					
search_ptlo	10408	10519/M	10519	10519/S	10519/S	10519/S	10519	10519/S	10519
		10519/S	10519	10519/S	10519/S	10519/S	10519	10519	10519
		10519	10519/M	10519	10550/M	10550	10550/S	10550/S	10550/S
		10550	10550/S	10550	10550/S	10550	10550/S	10550/S	10550/S
		10550	10550	10550	10550	10550/M	10550	10579/M	10579
		10579/S	10579/S	10579/S	10579	10579/S	10579	10579/S	10579
		10579/S	10579/S	10579/S	10579	10579	10579	10579	10579/M
		10579							
search_ptlo	10879	10925/M	10925	10925/S	10925/S	10925/S	10925	10925/S	10925/S
		10925/S	10925	10925	10925	10925	10925/M	10925	10925/M
		10925	10925/S	10925/S	10925	10925/M	10925/S	10925	
search_ptlo	10879	10925/M	10925	10925/S	10925/S	10925/S	10925	10925/S	10925/S
		10925	10925/S	10925/S	10925	10925	10925	10925	10925/M
		10925	10925/M	10925	10925/S	10925/S	10925	10925/M	10925/S
		10925							
search_ptlo	10879	10925/M	10925	10925/S	10925/S	10925/S	10925	10925/S	10925
		10925/S	10925	10925/S	10925/S	10925/S	10925	10925	10925
		10925	10925/M	10925					
search_ptlo	11095	11139/M	11139	11139/S	11139/S	11139/S	11139	11139/S	11139/S
		11139/S	11139	11139	11139	11139	11139/M	11139	11139/M
		11139	11139/S	11139/S	11139	11139/M	11139/S	11139	11151/M
		11151	11151/S	11151/S	11151/S	11151	11151/S	11151/S	11151/S
		11151	11151	11151	11151	11151/M	11151	11151/M	11151
		11151/S	11151/S	11151	11151/M	11151/S	11151		
search_ptlo	11095	11139/M	11139	11139/S	11139/S	11139/S	11139	11139/S	11139/S
		11139	11139/S	11139/S	11139	11139	11139	11139	11139/M
		11139	11139/M	11139	11139/S	11139/S	11139	11139/M	11139/S
		11139	11151/M	11151	11151/S	11151/S	11151/S	11151	11151/S
		11151/S	11151	11151/S	11151/S	11151	11151	11151	11151
		11151/M	11151	11151/M	11151	11151/S	11151/S	11151	11151/M
		11151/S	11151						
search_ptlo	11095	11139/M	11139	11139/S	11139/S	11139/S	11139	11139/S	11139
		11139/S	11139	11139/S	11139/S	11139/S	11139	11139	11139
		11139	11139/M	11139	11151/M	11151	11151/S	11151/S	11151/S
		11151	11151/S	11151	11151/S	11151	11151/S	11151/S	11151/S
		11151	11151	11151	11151	11151/M	11151		
search_xcb	6863	7054/M	7054	7054/M	7054				
search_xcb	6863	7056/M	7056	7056/M	7056	7056			
search_xcb	6863	7058/M	7058						
search_xcb	7085	7106/M	7108	7149/M	7151				
search_xcb	7197	7218/M	7220	7258/M	7260	7261			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
search_xcb	7305	7320/M 7322
search_xcb	7766	7866/M 7866 7866/M 7866 7898/M 7898 7898/M 7898
search_xcb	7766	7866/M 7866 7866/M 7866 7866 7898/M 7898 7898/M
		7898 7898
search_xcb	7766	7866/M 7866 7898/M 7898
search_xcb	7932	7979/M 7979 7979/M 7979
search_xcb	7932	7979/M 7979 7979/M 7979 7979
search_xcb	7932	7979/M 7979
search_xcb	8110	8172/M 8172 8172/M 8172
search_xcb	8110	8172/M 8172 8172/M 8172 8172
search_xcb	8110	8172/M 8172
search_xcb	8182	8227/M 8227 8227/M 8227
search_xcb	8182	8227/M 8227 8227/M 8227 8227
search_xcb	8182	8227/M 8227
search_xcb	8381	8499/M 8499 8499/M 8499
search_xcb	8381	8499/M 8499 8499/M 8499 8499
search_xcb	8381	8499/M 8499
search_xcb	8581	8630/M 8630 8630/M 8630
search_xcb	8581	8630/M 8630 8630/M 8630 8630
search_xcb	8581	8630/M 8630
search_xcb	9349	9406/M 9406 9406/M 9406
search_xcb	9349	9406/M 9406 9406/M 9406 9406
search_xcb	9349	9406/M 9406
search_xcb	9446	9479/M 9479 9479/M 9479
search_xcb	9446	9479/M 9479 9479/M 9479 9479
search_xcb	9446	9479/M 9479
search_xcb	9584	9623/M 9623 9623/M 9623 9647/M 9647 9647/M 9647
search_xcb	9584	9623/M 9623 9623/M 9623 9623 9647/M 9647 9647/M
		9647 9647
search_xcb	9584	9623/M 9623 9647/M 9647
search_xcb	10157	10201/M 10201 10201/M 10201
search_xcb	10157	10201/M 10201 10201/M 10201 10201
search_xcb	10157	10201/M 10201
search_xcb	10408	10519/M 10519 10519/M 10519 10550/M 10550 10550/M 10550
		10579/M 10579 10579/M 10579
search_xcb	10408	10519/M 10519 10519/M 10519 10519 10550/M 10550 10550/M
		10550 10550 10579/M 10579 10579/M 10579 10579
search_xcb	10408	10519/M 10519 10550/M 10550 10579/M 10579
search_xcb	10879	10925/M 10925 10925/M 10925
search_xcb	10879	10925/M 10925 10925/M 10925 10925
search_xcb	10879	10925/M 10925
search_xcb	11095	11139/M 11139 11139/M 11139 11151/M 11151 11151/M 11151
search_xcb	11095	11139/M 11139 11139/M 11139 11139 11151/M 11151 11151/M
		11151 11151
search_xcb	11095	11139/M 11139 11151/M 11151
seg	5242	5249/M 5252/M 5258
seg	5286	5293/M 5293/M 5293
seg	8182	8222/M 8222/M 8222 8232/M 8232/M 8232
segment_lock	3885	8223/P 8233 8249/P
segnum	5568	5571
segnum	8182	8221
segnum	8212	8220/M 8221/P
segnum	8412	8437/M 8443 8447/S

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
seqno	103	6492 6531 6550 6635/M 7452 7502 7532 7574
		7807 7957 8072 8137/M 8138 8266 8269/M 8445/M
		8608/M 9102 10784/M 11006 11019/M
sequence_number	4932	6492 6531 6550 6627/M 6627 6635 7452 7502
		7532 7574 7807 7957 8072 8137 8138 8266
		8269 8445/M 8445 8445 8608/M 8608 8608 9102
		10784 11006/M 11019
service_accumulator_since_swap	1316	9199/M
service_class	1326	6884/S 7824/S 7866/S 7898/S 7979/S 8172/S 8227/S 8496/S
		8499/S 8504/S 8505/S 8549/S 8551/S 8620/M 8624/S 8630/S
		9198/S 9406/S 9479/S 9623/S 9647/S 10045/S 10056/S 10193/S
		10201/S 10277/S 10329/S 10336/S 10373/S 10496/S 10519/S 10550/S
		10579/S 10925/S 11045 11110/S 11139/S 11151/S
service_class	10976	11045/M 11046/S 11047/S
service_class_counts	6219	8504/M 8505 8549/M 8550 11046/M 11047
service_class_p	9094	9198/M 9200 9201 9205
service_limit	1465	10045 10058 10283 10286 10388 10389 10396
service_remaining	1453	8739 8739 8739/M 8739 9712 9716 9723/M 9724
		10004/M 10030/M 10044/M 10068/M 10389/M 10466 10466 10466/M
		10466
service_remaining	4905	10059 10069
service_used	9948	9996/P 9997/P 10063/P 10064/P
service_used	10264	10272/M 10274/M 10293/M 10293
service_used	10371	10376/M 10385 10385 10388 10390 10396/M 10396
set_defined	1463	10058 10282 10386
sfd_p	1295	8466/M
sfid	2101	8232/P
sfid	2735	8222/P
sfid	5234	5246 5247 5248 5259
sfid	5286	5293/P
sfid	5286	5293 5293 5293 5293
sfid	8182	8222/P 8232/P
sfid	8182	8222 8222 8222 8222 8232 8232 8232 8232
sft$counter	1675	1644 1645 2260 2262 2263 2265
sft$file_space_limit_kind	2984	2742
signal	5025	8717/P
signal_interval	2252	8739 9744 10466
source	5599	5609
source	9936	10106
special_trap_count	2518	8724
st	2680	8427/P
st	5924	8447/M
stack_pages_saved	2537	7588
start_index	7651	7688/M 7693 7708/M
state	7622	7658/M 7660/M 7661/M 7662/M 7665/P 7666/M 7666/S 7669/M
		7671/M 7673/M 7674/M 7675/M 7676/M 7683 7683 7685/P
		7686/M 7688 7691 7696/M 7697/M 7698/P 7699/M 7701/M
		7702/M 7703/M 7714 7715 7716/P 7717/M 7721 7721/S
		7723/M 7723/S
state	8779	8840/P 8846/P
state	8891	8917/P 8923/P
state	10168	10171/P 10207/P
state	10231	10234/P 10243/P

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
statistics	1300	7051	7866	7879/M	7879	7884/M	7884	7889/M	7889
		7894/M	7894	7898	7970/M	7970	7975/M	7975	7979
		8163/M	8163	8168/M	8168	8172	8224/M	8224	8227
		8250/M	8250	8468/M	8469/M	8499	8630	8642/M	8642
		8643/M	8643	8729/M	8729	8730/M	8730	8739/M	8739
		8739/M	8739	8739	8739	9139	9249	9406	9479
		9623	9647	9685/M	9685	9687/M	9687	9742	9742
		10201	10274	10274	10466/M	10466	10466/M	10466	10466
		10466	10519	10550/M	10550	10550/M	10550	10550	10550/M
		10550	10559/M	10559	10565/M	10565	10567	10578	10579
		10910/M	10910	10917/M	10917	10925	10927/M	10927	11122/M
		11122	11128/M	11128	11130	11139	11151		
status	4856	9102/P	9103						
status	4868	9920/M							
status	4877	9950/M	9976/P	10088/P	10091/P	10094/P	10097/P		
status	4935	6493	6532	6551	6623/M	6648/M	6968/M	6991/M	7452
		7502	7533	7574	7587	7807	7848	7848	7853
		7866/M	7866/M	7869	7876	7878	7882	7882	7890/M
		7895/M	7898/M	7898/M	7909	7913	7957	7963	7964
		7971/M	7976/M	7979/M	7979/M	8072	8138	8151	8151
		8156	8157	8164/M	8169/M	8172/M	8172/M	8225/M	8227/M
		8227/M	8251/M	8266	8285	8287/M	8445/M	8499/M	8499/M
		8544/M	8608/M	8616/M	8630/M	8630/M	8741/M	9102	9259
		9260	9387	9388	9389	9390	9400	9401/M	9403
		9406/M	9406/M	9416	9418/M	9473	9474/M	9477	9479/M
		9479/M	9520	9521	9522	9523	9527	9529/M	9554
		9555	9556	9557	9561	9563/M	9617	9618/M	9621
		9623/M	9623/M	9641	9642/M	9645	9647/M	9647/M	10197
		10201/M	10201/M	10514/M	10519/M	10519/M	10550	10550	10550
		10550/M	10550	10550/M	10550/M	10550/M	10557/M	10560	10563
		10563	10564	10576/M	10579/M	10579/M	10581/M	10781/M	10869
		10903	10903	10909	10912/M	10916	10920/M	10925/M	10925/M
		11000/M	11086/M	11120/M	11123	11126	11126	11127	11138/M
		11139/M	11139/M	11148/M	11151/M	11151/M			
status	4986	8428/P	8429	8445/P	8446				
status	5005	8608/P	8609						
status	5023	8702/M	8704/P	8717/P	8718				
status	5647	5650/M	5651/M						
status	5676	5698/P							
status	5714	5719/P							
status	6464	6496/M	6496/M						
status	6467	6489/M	6496/P						
status	6505	6534/M	6534/M						
status	6508	6529/M	6534/P						
status	6567	6604/M	6604/M						
status	6571	6601/M	6604/P						
status	7416	7452/M	7452/P						
status	7416	7452/M	7452/M	7461/M	7461/M				
status	7416	7459/P							
status	7420	7449/M	7452/P	7453	7461/P				
status	7472	7505/P							
status	7517	7539/P							
status	7562	7577/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	7618	7668/P	7698/P						
status	8381	8445/M	8445/P						
status	8381	8445/M	8445/M						
status	8581	8608/M	8608/P						
status	8581	8608/M	8608/M						
status	8682	8704/M	8704/M						
status	8682	8739/P							
status	8780	8855/P							
status	9057	9102/M	9102/P						
status	9057	9102/M	9102/M						
status	9327	9333/P	9334						
status	9349	9373/M	9373/M	9412/M	9412/M	9433/M	9433/M		
status	9352	9365/M	9373/P	9412/P	9433/P				
status	9491	9543/M	9543/M	9577/M	9577/M				
status	9493	9502/M	9543/P	9577/P					
status	9586	9595/M							
status	9678	9745/P							
status	9936	9976/M	9976/M						
status	10408	10466/P							
status	10944	10978/M	10986/M	10987/M					
ste	2667	8427/P	8447/M						
stlc_allocation	2517	8853	8854/M						
str1	5604	5609/M	5611						
str1	9936	10106/M	10106						
str2	5605	5610/M	5611/M	5612					
str2	9936	10106/M	10106/M	10106					
subcode	4878	7814/M	9952						
subpriority	3792	10597/M							
subreq	10946	10979							
subsystem_give_up_cpu	2528	6917/M	6925/M	7866/M	7866/M	7898/M	7898/M	7979/M	7979/M
		8172/M	8172/M	8227/M	8227/M	8499/M	8499/M	8630/M	8630/M
		8821	8823/M	8905	8907/M	9406/M	9406/M	9479/M	9479/M
		9623/M	9623/M	9647/M	9647/M	10201/M	10201/M	10490/M	10519/M
		10519/M	10550/M	10550/M	10579/M	10579/M	10925/M	10925/M	11139/M
		11139/M	11151/M	11151/M					
subsystem_lock_priority_count	2529	6911	6923/M	6923	7863	7866	7866/M	7866	7898
		7898/M	7898	7979	7979/M	7979	8172	8172/M	8172
		8227	8227/M	8227	8499	8499/M	8499	8622/M	8630
		8630/M	8630	8849/M	8926/M	9406	9406/M	9406	9479
		9479/M	9479	9623	9623/M	9623	9647	9647/M	9647
		10201	10201/M	10201	10486	10487/M	10487	10519	10519/M
		10519	10550	10550/M	10550	10579	10579/M	10579	10925
		10925/M	10925	11139	11139/M	11139	11151	11151/M	11151
subsystem_locks_set	4955	7836	8850/M	8927/M	9426/M				
swap_queue_link	1275	11032/M	11033/M	11034/M					
swap_status	1269	5693	7459	7505	7539	7577	7668	7698	8465/M
		11031/M							
swapout_reason	9351	9392							
swapping_keypoints	4771	8472							
syc$mf_system_debugger	3364	6614	8445	8608	10847	10871			
syc$rc_update_job_task_enviro	170	7813							
syc$recover_pt1	10959	10947	10980						
syc$ucr_condition	3455	3466							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
syc$user_defined_condition	3456	3468							
system_breakpoint_selected	1296	6613	8441/M	8445	8608				
system_flags	2523	9024/M	9024	9158/M	9158	10131/M	10131	10797/M	10797
system_flags	4940	6628/M	8445/M	8608/M	9024	9027/M	9158	9160/M	10132
		10133/M	10796	10797	10798/M	11005/M			
system_give_up_cpu	2527	6928/M	7866/M	7898/M	7979/M	8172/M	8227/M	8499/M	8630/M
		8805	8814/M	8821	8824/M	8902	8903/M	8905	8908/M
		9406/M	9479/M	9623/M	9647/M	10201/M	10485/M	10519/M	10550/M
		10579/M	10925/M	11139/M	11151/M				
system_locks_set	10477	10477	10492	10532					
system_supplied_name	1264	7817	8473	9974					
system_supplied_name	4890	7817/M	9974						
system_table_lock_count	2522	6909	6910	7837	7861	7862	7866	7866	7898
		7898	7979	7979	8172	8172	8227	8227	8499
		8499	8630	8630	8708	8765/M	8765	8788	8792
		8802/M	8802	8806	8807	8820	8900	8901	8904
		9033	9041	9113	9121	9256	9386	9389	9406
		9406	9425	9479	9479	9519	9522	9553	9556
		9623	9623	9647	9647	10196	10201	10201	10479
		10482	10519	10519	10550	10550	10579	10579	10925
		10925	11139	11139	11151	11151			
system_task_id	2514	9516	9517	9518	9550	9551	9552	9613	9614
		9615	9637	9638	9639				
syt$180_idle_code	3693	1164	5795	5797					
syt$debug_control	4749	6350							
syt$job_recovery_subreq	10959	10946							
syt$monitor_flag	3362	3347	5752						
syt$monitor_flags	3347	2510	4939	6614	6616	8445	8445	8608	8608
		9020	9026	9141	9154	9159	10134	10790	10794
		10847	10871	11004					
syt$monitor_request_code	118	94	612	4711	4842	4855	4867	4876	4968
		4977	4985	4996	5004	5014	5022	6011	10943
syt$monitor_status	273	95	4712	4843	4856	4868	4877	4969	4978
		4986	4997	5005	5015	5023	5465	5647	5714
		5736	5746	5753	5759	6012	6467	6508	6571
		7420	8780	9327	9352	9493	9586	9678	10944
syt$perf_keypoints_enabled	4768	4764							
syt$ptl_recovery_info	10953	10949							
syt$rb_job_recovery	10942	10965							
syv$all_jobs_selected_for_debug	6351	8440							
syv$perf_keypoints_enabled	4764	8472							
tail	2171	6631/M	7877/M	7877	7877/M	7877	7877	7877/S	7877/M
		8074	8075	8082	8084/M	8086/S	8087/M	8087	8088/M
		8142	8144/M	8147/M	8223/M	8223	8223/M	8223	8223
		8223/S	8223/M	8249/M	8249	8249/M	8249	8249	8249/S
		8249/M	8288/M	8288/M	8288	8288/M	8288/M	8325/M	8343/M
		8344	8345/M	8347/M	8363/M	8363	8365/M	8365	8368
		8371/S	8375/M	8445/M	8608/M	10550/M	10550/M	10561/M	10561/M
		10561	10561/M	10561/M	10901/M	10933/M	11009/M	11124/M	11124/M
		11124	11124/M	11124/M					
tail_timeslice_insert	6873	6899	7040	7057					
tail_timeslice_insert	7766	7866	7866	7866	7898	7898	7898		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tail_timeslice_insert	7932	7979	7979	7979					
tail_timeslice_insert	8110	8172	8172	8172					
tail_timeslice_insert	8182	8227	8227	8227					
tail_timeslice_insert	8381	8499	8499	8499					
tail_timeslice_insert	8581	8630	8630	8630					
tail_timeslice_insert	9349	9406	9406	9406					
tail_timeslice_insert	9446	9479	9479	9479					
tail_timeslice_insert	9584	9623	9623	9623	9647	9647	9647		
tail_timeslice_insert	10157	10201	10201	10201					
tail_timeslice_insert	10408	10519	10519	10519	10550	10550	10550	10579	10579
		10579							
tail_timeslice_insert	10879	10925	10925	10925					
tail_timeslice_insert	11095	11139	11139	11139	11151	11151	11151		
task_created_after_last_swap	1309	8644/M	8731/M						
task_has_terminated	2516	8743/M							
task_id	96	7903	9006	10452/P					
task_index	11073	11075/M	11076	11076	11077/S	11078/S	11085/S	11086/P	11088/M
task_is_terminating	2525	10128/M							
task_list_p	10949	10984/S	10996	11002	11003	11006			
task_priority_changed	7803	7834/M	7840/M	7861					
task_queue	4011	8223/P	8233	8249/P					
task_rb	8213	8219/M	8220						
task_status	7731	7749	7753	7756					
task_status	7934	7960	7963	7964					
task_status	8046	8078	8099	8099	8102				
task_time_slice_used	8682	8739/M	8739/P	8739	8739	8739	8739	8739	
task_time_slice_used	9679	9690/M	9694/P	9712	9716	9724	9725	9726	
task_time_slice_used	10408	10466/M	10466/P	10466	10466	10466	10466	10466	
taskid	1152	7587	7749/S	7756/S	8018/S	8019/S	8022/S	8023/S	8096
		8533/S	8536/S	8544/P	8614/S	8647	8741/P	8742/P	8753/S
		8803	8896	9018/S	9134/S	10080/S	10130/S	10132/S	10133/S
		10134/S	10472	10746/M	10783/M	10784/M	11116		
taskid	5008	8608/P	8615/S	8616/S	8630/P	8635/P	8646		
taskid	6465	6491	6491	6492/S	6492	6493/S			
taskid	6506	6530	6530	6531/S	6531	6532/S			
taskid	6545	6549	6549	6550/S	6550	6551/S			
taskid	6570	6634/M	6635/M						
taskid	7353	7369/S	7370						
taskid	7376	7403							
taskid	7416	7452	7452	7452/S	7452	7452/S			
taskid	7417	7452/P	7458/S	7459/S	7463/S				
taskid	7472	7502	7502	7502/S	7502	7502/S			
taskid	7473	7502/P	7503/S	7505/S	7509/S				
taskid	7518	7531	7531	7532/S	7532	7533/S	7537/S	7539/S	7543/S
taskid	7562	7574	7574	7574/S	7574	7574/S			
taskid	7563	7574/P	7575/S	7577/S	7581/S	7587/S	7587		
taskid	7766	7807	7807	7807/S	7807	7807/S			
taskid	7767	7807/P	7808/S	7812	7864/P	7866/P	7898/P	7903	7914/P
taskid	7932	7957	7957	7957/S	7957	7957/S			
taskid	7933	7957/P	7958/S	7979/P					
taskid	8044	8072	8072	8072/S	8072	8072/S			
taskid	8045	8068/S	8072/P	8075	8083	8084	8086	8088	8096
taskid	8110	8138	8138	8138/S	8138	8138/S			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
taskid	8112	8135/M	8136/S	8137/M	8138/P	8172/P			
taskid	8183	8227/P	8235						
taskid	8259	8266	8266	8266/S	8266	8266/S			
taskid	8260	8266/P	8267/S	8268/M	8269/M				
taskid	8381	8445/M	8445/M						
taskid	8581	8608/M	8608/M						
taskid	8581	8635/S	8635						
taskid	8682	8742							
taskid	9057	9102	9102	9102/S	9102	9102/S			
tasks_not_in_long_wait	1647	7879/M	7879	7884/M	7884	8469/M	8643/M	8643	8730/M
		8730	9139	10550/M	10550	10550/M	10550	10565/M	10566
		10567	10910/M	10910	10917/M	10917	11128/M	11129	11130
tasks_not_swappable_count	9361	9382/M	9393/M	9393	9432				
tasks_not_swappable_count	9500	9510/M	9525/M	9525	9541	9559/M	9559	9575	
temp_next_cyclic_aging	8682	8754/M	8754	8754					
temp_next_cyclic_aging	9282	9303/M	9304	9307					
temp_next_cyclic_aging	9362	9435/M	9436	9439					
temp_next_cyclic_aging	9879	9888/M	9889	9892					
temp_next_cyclic_aging	10408	10530/M	10530	10530					
temp_xcb_p	8781	8840/P	8841	8841	8842/S	8844/S	8846/P		
time	10433	10748/M	10750/M	10752	10753/M	10753	10754	10755/M	10756
		10757/M	10759						
time_left_in_interval	4796	8739	8739/M	8739	9694	9694/M	9694	9761	9771/M
		9772	10466	10466/M	10466	10774	10774/M	10774	
time_next_scan_wait_not_queued	8277	8286							
time_spent_in_job_mode	1630	8739/M	8739	8739/M	8739	8739/M	8739	8739	9687/M
		9688	9701/M	9702	9709/M	9709	9742	9930	10274
		10466/M	10466	10466/M	10466	10466/M	10466	10466	
time_spent_in_mtr_mode	1631	8739/M	8739	8739/M	8739	8739/M	8739	8739	9685/M
		9686	9698/M	9699	9707/M	9707	9743	9921	10275
		10466/M	10466	10466/M	10466	10466/M	10466	10466	
time_used	8682	8739	8739	8739	8739	8739	8739		
time_used	9661	9694	9694	9694	9694	9694	9694		
time_used	9754	9761	9772	9776	9786	9793	9811		
time_used	10408	10466	10466	10466	10466	10466	10466	10774	10774
		10774	10774	10774	10774				
timeslice	2543	6893	6895	6897/M	6900/M	6901/M	7866	7866	7866/M
		7866/M	7866/M	7898	7898	7898/M	7898/M	7898/M	7979
		7979	7979/M	7979/M	7979/M	8172	8172	8172/M	8172/M
		8172/M	8227	8227	8227/M	8227/M	8227/M	8496/M	8499
		8499	8499/M	8499/M	8499/M	8624/M	8630	8630	8630/M
		8630/M	8630/M	8739	8739	8739/M	8739/M	8739/M	8739/M
		9406	9406	9406/M	9406/M	9406/M	9479	9479	9479/M
		9479/M	9479/M	9623	9623	9623/M	9623/M	9623/M	9647
		9647	9647/M	9647/M	9647/M	9725	9726	9728/M	9730/M
		9733/M	9735/M	10193/M	10201	10201	10201/M	10201/M	10201/M
		10466	10466	10466/M	10466/M	10466/M	10466/M	10519	10519
		10519/M	10519/M	10519/M	10550/M	10550/M	10550	10550	10550/M
		10550/M	10550/M	10579	10579	10579/M	10579/M	10579/M	10786
		10922/M	10923/M	10925	10925	10925/M	10925/M	10925/M	11110/M
		11139	11139	11139/M	11139/M	11139/M	11151	11151	11151/M
		11151/M	11151/M						
timeslice	6879	6884/M	6893	6895	6897	6900	6901		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
timeslice	7766	7866/M	7866	7866	7866	7866	7866	7898/M	7898
		7898	7898	7898	7898				
timeslice	7932	7979/M	7979	7979	7979	7979	7979		
timeslice	8110	8172/M	8172	8172	8172	8172	8172		
timeslice	8182	8227/M	8227	8227	8227	8227	8227		
timeslice	8381	8499/M	8499	8499	8499	8499	8499		
timeslice	8581	8630/M	8630	8630	8630	8630	8630		
timeslice	9349	9406/M	9406	9406	9406	9406	9406		
timeslice	9446	9479/M	9479	9479	9479	9479	9479		
timeslice	9584	9623/M	9623	9623	9623	9623	9623	9647/M	9647
		9647	9647	9647	9647				
timeslice	10157	10201/M	10201	10201	10201	10201	10201		
timeslice	10408	10519/M	10519	10519	10519	10519	10519	10550/M	10550
		10550	10550	10550	10550	10579/M	10579	10579	10579
		10579	10579						
timeslice	10879	10925/M	10925	10925	10925	10925	10925		
timeslice	11095	11139/M	11139	11139	11139	11139	11139	11151/M	11151
		11151	11151	11151	11151				
tmc$	5038	5044	5047	5050	5053	5056	5059	5062	5065
		5068	5071	5074	5077	5080	5083	5086	5089
tmc$broken_task_fault_id	3383	3433							
tmc$btc_invalid_a0	3481	3502							
tmc$btc_invalid_p	3481	3502							
tmc$btc_mcr_traps_disabled	3482	3503							
tmc$btc_mf_traps_disabled	3481	3501							
tmc$btc_mntr_fault_buffer_full	3480	3501							
tmc$btc_system_error	3483	3497							
tmc$btc_ucr_traps_disabled	3482	3503							
tmc$cpo_interactive_command	4914	7815	10035	10040					
tmc$cpo_interrupt_restore	4915	10049							
tmc$cpo_operator	4910	9981	9993						
tmc$cpo_recovery	4916	9981							
tmc$cpo_save_swap_file_sfid	4912	10035							
tmc$cpo_set_job_unswappable	4913	10022							
tmc$cpo_user	4911	10013							
tmc$cycle_reason	619	613	6376						
tmc$dummy_fault	3384	3439							
tmc$flag_available_31	3618	3622							
tmc$fnx_continue	4824	7657	8846/P	8923/P	10207/P	10243/P			
tmc$fnx_job	4824	7670	8839/P	8917/P	10005/P	10020/P	10033/P	10047/P	10071/P
		10393/P							
tmc$fnx_system	4824	7659	7683						
tmc$ir_dm_system_tasks	4837	9512	9609	10094/P	10097/P				
tmc$ir_other_system_tasks	4837	10088/P	10091/P						
tmc$is_idle_initiated	3779	8614	8753	9394	9526	9560	10527		
tmc$is_idled	3779	7837	7888	7893	7969	7974	8162	8167	8217
		8533	8754	8754	9298	9399	9422	9472	9533
		9567	9616	9640	9883	9883	10528	10530	10530
		10550	10575	10907	11007				
tmc$is_idled_sched_notified	3780	7891	7973	8165	8252	8615	9420	9531	9565
		10550	10582	10908					
tmc$is_not_idled	3779	6629	8445	8608	9409	9482	9626	9650	10080
		10569	11132						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$maximum_monitor_faults	3388	3379							
tmc$maximum_ptl	4922	6338	6344						
tmc$maximum_signals	3598	3595							
tmc$maximum_system_task_id	3631	3634							
tmc$mcr_fault	3383	3435							
tmc$opt_return	4950	6495	6533	7452/P	7452	9102/P	9102		
tmc$rc_ready_conditional_wi	3782	7921	9195/P	10453/P	11020/P				
tmc$signal_available_63	3580	3591							
tmc$stid_administer_log	3640	9516	9550	9613	9637				
tmc$stid_dm_split_al	3641	9517	9551	9614	9638				
tmc$stid_null_task	3637	3634							
tmc$stid_volume_space_managemnt	3642	9518	9552	9615	9639				
tmc$ts_executing	3770	7587	7848	7964	8157	9387	9520	9554	10781
tmc$ts_first_external_queue	3757	7749	8078	8156	8157				
tmc$ts_first_ready_uncond	3756	7849							
tmc$ts_first_status_in_wait_q	3753	9259							
tmc$ts_io_wait_not_queued	3774	3756	7753	7960					
tmc$ts_io_wait_queued	3777	8099	8151	8152					
tmc$ts_last_status_in_dct	3752	9403	9416	9477	9527	9561	9621	9645	10197
		10550	10556	10903	11119				
tmc$ts_last_status_in_wait_q	3754	7876	9260	9523	9557	10550	10560	10903	11123
tmc$ts_null	3761	6493	6532	6551	6648	7452	7502	7533	7574
		7750	7807	7850	7957	7967	8018	8072	8078
		8138	8160	8266	8544	8741	9102	9154	9161
		9164	9167	9181	10480	10558	10869	11086	11121
tmc$ts_page_wait	3776	3757	8099	8151	8152				
tmc$ts_ready	3761	3752	6623	6968	7853	7866	7895	7898	7976
		7979	8169	8172	8225	8227	8445	8499	8608
		8630	9401	9406	9474	9479	9618	9623	9642
		9647	10201	10514	10519	10550	10550	10576	10579
		10920	10925	11138	11139	11148	11151		
tmc$ts_ready_and_selected	3763	6991	7848	7866	7898	7979	8172	8227	8499
		8630	9388	9406	9479	9521	9555	9623	9647
		10201	10519	10550	10579	10925	11139	11151	
tmc$ts_ready_but_swapped	3772	7869	7890	7909	7971	8164	8251	8616	9400
		9418	9473	9529	9563	9617	9641	10550	10581
		10912							
tmc$ts_segment_lock_wait	3776	7913							
tmc$ts_timed_wait_not_queued	3768	3755	7883	8285	9038	9118	9138	10564	11127
tmc$ts_timeout_reqexp_inflong	3771	9042	9122	11000					
tmc$ts_timeout_reqexp_infvlong	3771	7882	9044	9124	9136	9391	10564	11127	
tmc$ts_timeout_reqexp_longlong	3766	9034	9114	9390					
tmc$ts_timeout_reqexp_longvlong	3766	3754	7878	8287	9036	9116	9137	10550	10550
		10563	10909	10916	11126				
tmc$ts_timeout_reqexp_shortshrt	3765	3753	8019	8022	8929	9523	9557		
tmc$ujte_dispatching_priority	4878	4887	7814	9968					
tmc$ujte_expand_ptl	4878	4892	10099						
tmc$ujte_idle_dm_sys_tasks	4880	4895	10093						
tmc$ujte_idle_other_sys_tasks	4879	4894	10087						
tmc$ujte_restart_dm_systasks	4880	4895	10096						
tmc$ujte_restart_other_systasks	4879	4894	10090						
tmc$ujte_set_non_swappable	4879	4894	10079						
tmc$ujte_set_task_terminating	4881	10123							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$ujte_update_debug_masks	4880	4895	10120						
tmc$ujte_xp_register	4878	4882	9953						
tmc$wi_null	4961	9180	10804						
tmc$wi_wait_inhibited	4961	7924	9143	9163	10801				
tmc$wi_wait_selected	4962	7922	9171						
tmc$wi_wait_selected_r3	4962	7923	9169	10848					
tme$invalid_global_taskid	5044	6496/P	6534/P	7452/P	8704/P	9102/P			
tme$job_swapped_out	5056	7461/P							
tme$ptl_full	5047	6604/P	8445/P	8608/P	10987				
tmk$monitor_base	839	5121	5124	5127	5130	5133	5136	5139	5142
		5145	5148	5151	5154	5157	5160	5164	5167
		5170	5173	5177	5181	5184	5187	5190	5193
		5196	5199	5202	5205	5208	5211	5214	5217
		5220	5223	5226					
tmk$switch_task	5173	10441	10524	10822					
tmp$assign_ptl	6567	6637	8445	8608					
tmp$calculate_dct_priority_int	9852	6694	6979	7014	7864	7866	7866	7898	7898
		7979	7979	8172	8172	8227	8227	8499	8499
		8630	8630	8739	8739	8739	9406	9406	9417
		9479	9479	9528	9562	9623	9623	9647	9647
		9694	9694	9694	9766	9781	9806	9868	10198
		10201	10201	10466	10466	10466	10519	10519	10550
		10550	10579	10579	10725	10774	10774	10774	10925
		10925	11139	11139	11151	11151			
tmp$cause_task_switch	8011	8041							
tmp$check_for_swapout_candidate	9217	9273							
tmp$check_ptl_lock	5668	5673							
tmp$check_taskid	6464	6503							
tmp$check_taskid_with_lock_set	6505	6540	7452	9102					
tmp$check_timed_wait_not_queued	8276	8294							
tmp$clear_lock	6444	6457	6494	6502	7454	7466	7512	7547	7599
		7725	7757	7927	7983	8039	8104	8176	8270
		8292	8451	8508	8553	8610	8677	8756	8870
		8961	9010	9028	9048	9104	9211	9250	9257
		9271	9339	9486	9539	9573	9631	9655	9975
		10137	10141	10210	10246	10523	10589	10820	10988
		11022	11155						
tmp$create_job	8381	8510							
tmp$create_task	8581	8679							
tmp$cycle	8875	8963							
tmp$dct_ready_task	6863	7064	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10518	10550	10579
		10925	11139	11151					
tmp$delay	8966	9055							
tmp$dequeue_task	8110	8178							
tmp$exit_job	8513	8578							
tmp$fetch_task_statistics	9900	9932							
tmp$find_next_queued_task	8259	8272							
tmp$find_next_xcb	7618	7727	8839	8846	8917	8923	10171	10207	10234
		10243							
tmp$find_xcb	7416	7468							
tmp$free_unrecovered_tasks	11068	11091							
tmp$get_top_of_stack	7562	7601							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
tmp$get_xcb_access_status	5676	5701	7459	7505	7539	7577	7668	7698
tmp$get_xcb_p	7472	7514						
tmp$get_xcb_p_from_ptlo	7607	7614	9255	10515	11149			
tmp$idle_non_dispatchable_job	9322	9341						
tmp$idle_tasks_in_job	9349	9333	9443					
tmp$idle_tasks_in_system_job	9491	9581	10088	10094				
tmp$insert_timed_wait_queue	8298	8288	8351	10561	11124			
tmp$job_recovery_requests	10964	11058						
tmp$mtr_begin_lock_activity	8761	8767						
tmp$mtr_end_lock_activity	8770	8872						
tmp$mtr_update_job_task_enviro	9936	7821	10143					
tmp$mtr_wait	9057	9214						
tmp$obtain_ijl_ordinal_from_ptl	6558	6564						
tmp$queue_task	8044	8106						
tmp$reissue_monitor_request	7988	8008	10083					
tmp$remove_task_from_dct	6660	6726	7864	9417	9528	9562	10198	10725
tmp$remove_task_from_q	8358	7877	8223	8249	8377			
tmp$reset_dispatching_control	10362	8739	9717	9997	10064	10404	10466	
tmp$restart_idled_tasks	9446	9488						
tmp$restart_tasks_in_system_job	9584	9658	10091	10097				
tmp$send_signal	5757	8717						
tmp$set_lock	6421	6439	6490	7450	7501	7529	7573	7653 7748
		7806	7956	8017	8070	8134	8265	8282 8444
		8532	8607	8727	8783	8897	9004	9100 9247
		9329	9469	9504	9607	9951	10170	10233 10461
		10593	10981	11107				
tmp$set_monitor_flag	5751	8855						
tmp$set_swapout_candidate	9278	9268	9312	10571	11134			
tmp$set_system_flag	5744	8739	9745	10466				
tmp$set_task_ready	7766	7929	9194	10452	11020			
tmp$set_task_ready_uncond	7932	7985						
tmp$set_task_wait	7730	7763						
tmp$set_up_debug_registers	10832	10854						
tmp$stop_if_bad_taskid	6544	6555	7502	7574	7807	7957	8072	8138 8266
tmp$switch_task	10408	10824						
tmp$switch_task_from_failing_cp	11095	11157						
tmp$task_exit	8682	8758						
tmp$test_get_xcb_p	7517	7549						
tmp$update_debug_registers	10863	10121	10875					
tmp$update_job_task_cpu_selects	10224	10248						
tmp$update_job_task_environment	10157	10005	10020	10033	10047	10071	10212	10393
tmp$update_system_task_list	5764	8713						
tmt$broken_task_condition	3480	3496						
tmt$broken_task_monitor_fault	3494	3434						
tmt$change_priority_origin	4910	4888						
tmt$cpu_execution_statistics	4780	6329						
tmt$dct_entry	5099	5106	6676	6746	6872	7072	7184	7293
tmt$dispatch_control	3723	1159						
tmt$dispatch_control_table	5106	6366						
tmt$dispatching_control_sets	4814	6333	9830	9855	10429			
tmt$dispatching_controls	4785	6332						
tmt$dispatching_prio_controls	4795	4790	6334					
tmt$dispatching_priority_time	4800	4797						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
tmt$dual_state_dispatch_prior	3786	6335	6335					
tmt$dual_state_priority_entry	3790	1146	3787					
tmt$find_next_xcb_state	4826	7622	8779	8891	10168	10231		
tmt$fnx_search_type	4824	4827	7619	10160	10227			
tmt$idle_resume_sys_task_kind	4837	9492	9585					
tmt$idle_status	3779	4937						
tmt$mcr_faults	3519	3436						
tmt$monitor_fault_buffer	3373	2547						
tmt$monitor_fault_buffers	3379	3374	3375	3376				
tmt$monitor_fault_identifiers	3382	3432	3508					
tmt$option	4950	6466	6507					
tmt$primary_task_list	4948	4893	6355					
tmt$primary_task_list_entry	4930	4948	6595	6679	6749	6874	6878	7801 7953
		8065	8131	8185	8263	8359	8997	9091 10165
		10435	10891	10975	11101			
tmt$ptl_flags	4954	4941						
tmt$ptl_lock	4734	6352	6352	6421	6444			
tmt$rb_cycle	611	8876						
tmt$rb_delay	4841	8600	8967					
tmt$rb_exit_job	4995	8514						
tmt$rb_fetch_task_statistics	4866	9901						
tmt$rb_initiate_job	4984	8382						
tmt$rb_initiate_task	5003	8582						
tmt$rb_ready_task	93	83						
tmt$rb_task_exit	5021	8683						
tmt$rb_update_job_task_enviro	4875	7800	9937					
tmt$rb_wait	4854	4862						
tmt$rb_wait_signal	4862	9058						
tmt$ready_condition	3782	7769						
tmt$signal	3536	3531						
tmt$signal_buffer	3528	2548						
tmt$signal_buffers	3595	3529	3530	3531				
tmt$system_flags	3601	2523	4940	6628	8445	8608	9027	9160 10133
		10796	10798	11005				
tmt$system_task_id	3634	2514						
tmt$task_queue_link	2169	2138	4011	4938	6343	6343	8047	8111 8209
		8210	8360					
tmt$task_status	3761	3726	4935	4936	6377	7731	7934	8046 9090
tmt$time_limits	4802	4800						
tmt$wait_inhibited	4961	4956						
tmt$xcb_offset_size	4952	4933						
tmv$cpu_execution_statistics	6329	8739/M	8739	8739/M	8739	9698/M	9699	9701/M 9702
		10466/M	10466	10466/M	10466			
tmv$cycle_delay_time	6347	8023	8928	8930				
tmv$dct	6366	6602	6608/M	6651	6652/M	6654/S	6656/M	6682 6724/M
		6752	6776/M	6969	6984/M	7004	7015/M	7062/M 7864
		7864/M	7866	7866/M	7866	7866/M	7866/M	7898 7898/M
		7898	7898/M	7898/M	7979	7979/M	7979	7979/M 7979/M
		8172	8172/M	8172	8172/M	8172/M	8227	8227/M 8227
		8227/M	8227/M	8445	8445/M	8499	8499/M	8499 8499/M
		8499/M	8544	8544/M	8544/S	8544/M	8608	8608/M 8630
		8630/M	8630	8630/M	8630/M	8741	8741/M	8741/S 8741/M
		9406	9406/M	9406	9406/M	9406/M	9417	9417/M 9479

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9479/M	9479	9479/M	9479/M	9528	9528/M	9562	9562/M
		9623	9623/M	9623	9623/M	9623/M	9647	9647/M	9647
		9647/M	9647/M	10113	10114/M	10116/S	10118/M	10198	10198/M
		10201	10201/M	10201	10201/M	10201/M	10519	10519/M	10519
		10519/M	10519/M	10550	10550/M	10550	10550/M	10550/M	10579
		10579/M	10579	10579/M	10579/M	10674	10725	10725/M	10925
		10925/M	10925	10925/M	10925/M	10997	10997/M	11086	11086/M
		11086/S	11086/M	11139	11139/M	11139	11139/M	11139/M	11151
		11151/M	11151	11151/M	11151/M				
tmv$dct_priority_integer	6330	6694	6979	7014	7760	7864	7866	7866	7898
		7898	7979	7979	8026	8094	8172	8172	8227
		8227	8499	8499	8561	8630	8630	8739	8739
		8739	8748	8935	9051	9176	9406	9406	9417
		9479	9479	9528	9562	9623	9623	9647	9647
		9694	9694	9694	9766	9781	9806	9866	10198
		10201	10201	10466	10466	10466	10519	10519	10541
		10550	10550	10579	10579	10725	10774	10774	10774
		10925	10925	11139	11139	11151	11151		
tmv$dedicate_a_cpu_to_nos	6364	6949	7866	7898	7979	8172	8227	8499	8630
		9406	9479	9623	9647	10201	10519	10550	10579
		10594	10747	10925	11139	11151			
tmv$dispatch_priority_integer	6331	6940	6988	7866	7866	7898	7898	7979	7979
		8172	8172	8227	8227	8499	8499	8630	8630
		8739	8739	8739	8817	8827	9406	9406	9479
		9479	9623	9623	9647	9647	9694	9694	9694
		9768	9782	9807	9841	10201	10201	10466	10466
		10466	10484	10489	10519	10519	10550	10550	10579
		10579	10774	10774	10774	10925	10925	11139	11139
		11151	11151						
tmv$dispatching_control_sets	6333	6692/M	6692	6694	6976/M	6976	6978	7012/M	7012
		7014	7864/M	7864	7864	7866/M	7866	7866	7866/M
		7866	7866	7898/M	7898	7898	7898/M	7898	7898
		7979/M	7979	7979	7979/M	7979	7979	8172/M	8172
		8172	8172/M	8172	8172	8227/M	8227	8227	8227/M
		8227	8227	8499/M	8499	8499	8499/M	8499	8499
		8630/M	8630	8630	8630/M	8630	8630	8739/M	8739/M
		8739/M	8739	8739	8739	8739/M	8739	8739	8739
		8739	8739/M	8739	8739/M	8739	8739	8739	9406/M
		9406	9406	9406/M	9406	9406	9417/M	9417	9417
		9479/M	9479	9479	9479/M	9479	9479	9528/M	9528
		9528	9562/M	9562	9562	9623/M	9623	9623	9623/M
		9623	9623	9647/M	9647	9647	9647/M	9647	9647
		9694/M	9694/M	9694/M	9694	9694	9694	9694/M	9694
		9694	9694	9694	9694/M	9694	9694/M	9694	9694
		9694	9763/M	9764/M	9765/M	9766	9768	9774	9778/M
		9778	9781	9782	9791	9795/M	9795	9802/M	9802
		9806	9807	9832	9857	10198/M	10198	10198	10201/M
		10201	10201	10201/M	10201	10201	10466/M	10466/M	10466/M
		10466	10466	10466	10466/M	10466	10466	10466	10466
		10466/M	10466	10466/M	10466	10466	10466	10519/M	10519
		10519	10519/M	10519	10519	10550/M	10550	10550	10550/M
		10550	10550	10579/M	10579	10579	10579/M	10579	10579
		10657	10725/M	10725	10725	10774/M	10774/M	10774/M	10774

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10774	10774	10774/M	10774	10774	10774	10774	10774/M
		10774	10774/M	10774	10774	10774	10925/M	10925	10925
		10925/M	10925	10925	11139/M	11139	11139	11139/M	11139
		11139	11151/M	11151	11151	11151/M	11151	11151	
tmv$dispatching_control_time	6334	8739	8739/M	8739/M	8739	8739	8739/M	8739	8739
		8739/M	8739	9694	9694/M	9694/M	9694	9694	9694/M
		9694	9694	9694/M	9694	9761	9762/M	9771/M	9771
		9776	9784/M	9785	9793	9809/M	9810	10466	10466/M
		10466/M	10466	10466	10466/M	10466	10466	10466/M	10466
		10774	10774/M	10774/M	10774	10774	10774/M	10774	10774
		10774/M	10774						
tmv$dispatching_controls	6332	8739	8739	8739	8739	8739	9694	9694	9694
		9694	9694	9760	9762	9763	9790	9801	10466
		10466	10466	10466	10466	10774	10774	10774	10774
		10774							
tmv$dual_state_dispatch_prior	6335	10494	10788						
tmv$io_wait_task_count	6338	7754/M	7754	7961/M	7961	8100/M	8100	8153/M	8153
		10736							
tmv$long_wait_force_swap_time	6346	9267							
tmv$long_wait_swap_time	6345	9033	9041	9113	9121				
tmv$multiple_cpus_active	6339	10596							
tmv$null_global_task_id	5918	6491	6530	6549	7056	7260	7452	7502	7531
		7574	7807	7827	7866	7898	7957	7979	8072
		8138	8172	8227	8266	8499	8630	9101	9102
		9187	9406	9479	9623	9647	10041	10201	10519
		10550	10579	10925	11139	11151			
tmv$ptl_lock	6352	5671	6490/P	6494/P	6502/P	7450/P	7454/P	7466/P	7501/P
		7512/P	7529/P	7547/P	7573/P	7599/P	7653/P	7725/P	7748/P
		7757/P	7806/P	7927/P	7956/P	7983/P	8017/P	8039/P	8070/P
		8104/P	8134/P	8176/P	8265/P	8270/P	8282/P	8292/P	8444/P
		8451/P	8508/P	8532/P	8553/P	8607/P	8610/P	8677/P	8727/P
		8756/P	8783/P	8870/P	8897/P	8961/P	9004/P	9010/P	9028/P
		9048/P	9100/P	9104/P	9211/P	9247/P	9250/P	9257/P	9271/P
		9329/P	9339/P	9469/P	9486/P	9504/P	9539/P	9573/P	9607/P
		9631/P	9655/P	9951/P	9975/P	10137/P	10141/P	10170/P	10210/P
		10233/P	10246/P	10461/P	10523/P	10589/P	10593/P	10820/P	10981/P
		10988/P	11022/P	11107/P	11155/P				
tmv$ptl_p	6355	6491	6492	6493	6530	6531	6532	6549	6550
		6551	6562	6607	6648/M	6649/M	6650/M	6654/M	6681
		6707	6712/M	6712	6751	6766	6771/M	6771	6881
		6965	7020	7021/M	7034	7035/M	7041/M	7054	7054
		7054	7054	7054	7054/M	7054	7054/M	7054	7054/M
		7054	7054	7054	7054	7054	7054	7054	7054
		7054/M	7054	7054/M	7054	7054/M	7054	7056	7056
		7056	7056	7056	7056/M	7056	7056	7056/M	7056
		7056/M	7056	7056	7056	7056	7056	7056	7056
		7056	7056	7056/M	7056	7056/M	7056	7056/M	7056
		7058	7058	7058	7058	7058	7058	7058	7058/M
		7058	7058/M	7058/M	7104	7105	7105	7107	7110
		7112/M	7112	7114/M	7114	7115/M	7133	7146	7148
		7148	7150	7155	7160	7163	7168/M	7168	7169/M
		7169	7170/M	7171	7216	7217	7217	7219	7222
		7224/M	7224	7227	7229/M	7229	7230/M	7243	7255

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

IDENTIFIER----------DEFINED----------REFERENCES
ON LINE

7257	7257	7259	7265	7269	7272	7275	7277/M
7277	7278/M	7278	7279/M	7280	7318	7319	7319
7321	7324	7327	7330	7333/M	7333	7335/M	7336/M
7369/M	7369	7370/M	7407	7410/M	7410	7411/M	7452
7452	7452	7458/P	7459/P	7463	7502	7502	7502
7503/P	7505/P	7509	7531	7532	7533	7537/P	7539/P
7543	7574	7574	7574	7575/P	7577/P	7581	7587
7612	7721	7723	7749	7756/M	7807	7807	7807
7808	7864	7864	7864/M	7864	7866	7866	7866
7866/M	7866	7866/M	7866/M	7866	7866	7866	7866
7866	7866/M	7866	7866/M	7866	7866/M	7866	7866
7866	7866	7866	7866	7866	7866	7866/M	7866
7866/M	7866	7866/M	7866	7866	7866	7866	7866
7866	7866/M	7866	7866	7866/M	7866	7866/M	7866
7866	7866	7866	7866	7866	7866	7866	7866
7866/M	7866	7866/M	7866	7866/M	7866	7866	7866
7866	7866	7866	7866	7866	7866/M	7866	7866/M
7866/M	7877/M	7877/M	7898	7898	7898	7898/M	7898
7898/M	7898/M	7898	7898	7898	7898	7898	7898/M
7898	7898/M	7898	7898/M	7898	7898	7898	7898
7898	7898	7898	7898	7898/M	7898	7898/M	7898
7898/M	7898	7898	7898	7898	7898	7898	7898/M
7898	7898	7898/M	7898	7898/M	7898	7898	7898
7898	7898	7898	7898	7898	7898	7898/M	7898
7898/M	7898	7898/M	7898	7898	7898	7898/M	7898
7898	7898	7898	7898/M	7898	7898/M	7898	7957
7957	7957	7958	7979	7979	7979	7979/M	7979
7979/M	7979/M	7979	7979	7979	7979	7979	7979/M
7979	7979/M	7979	7979/M	7979	7979	7979	7979
7979	7979	7979	7979	7979/M	7979	7979/M	7979
7979/M	7979	7979	7979	7979	7979	7979	7979/M
7979	7979	7979/M	7979	7979/M	7979	7979	7979
7979	7979	7979	7979	7979	7979	7979/M	7979
7979/M	7979	7979/M	7979	7979	7979	7979	7979
7979	7979	7979	7979/M	7979	7979/M	7979	8018
8019	8022/M	8023/M	8068	8072	8072	8072	8086/M
8136	8138	8138	8138	8147/M	8172	8172	8172
8172/M	8172	8172/M	8172/M	8172	8172	8172	8172
8172	8172/M	8172	8172/M	8172	8172/M	8172	8172
8172	8172	8172	8172	8172	8172	8172/M	8172
8172/M	8172	8172/M	8172	8172	8172	8172	8172
8172	8172/M	8172	8172	8172/M	8172	8172/M	8172
8172	8172	8172	8172	8172	8172	8172	8172
8172/M	8172	8172/M	8172	8172/M	8172	8172	8172
8172	8172	8172	8172	8172	8172/M	8172	8172/M
8172/M	8223/M	8223/M	8227	8227	8227	8227/M	8227
8227/M	8227/M	8227	8227	8227	8227	8227	8227/M
8227	8227/M	8227	8227/M	8227	8227	8227	8227
8227	8227	8227	8227	8227/M	8227	8227/M	8227
8227/M	8227	8227	8227	8227	8227	8227	8227/M
8227	8227	8227/M	8227	8227/M	8227	8227	8227
8227	8227	8227	8227	8227	8227	8227/M	8227
8227/M	8227	8227/M	8227	8227	8227	8227	8227

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

IDENTIFIER----------DEFINED----------REFERENCES
ON LINE

8227	8227	8227	8227/M	8227	8227/M	8227/M	8239
8249/M	8249/M	8266	8266	8266	8267	8269	8284
8285	8285	8287/M	8288	8288	8288	8288	8288/M
8288/M	8288/M	8288/M	8321	8327	8327	8330	8337/M
8341/M	8343/M	8347/M	8365/M	8371/M	8445	8499	8499
8499	8499/M	8499	8499/M	8499/M	8499	8499	8499
8499	8499	8499/M	8499	8499/M	8499	8499/M	8499
8499	8499	8499	8499	8499	8499	8499	8499/M
8499	8499/M	8499	8499/M	8499	8499	8499	8499
8499	8499	8499/M	8499	8499	8499/M	8499	8499/M
8499	8499	8499	8499	8499	8499	8499	8499
8499	8499/M	8499	8499/M	8499	8499/M	8499	8499
8499	8499	8499	8499	8499	8499	8499/M	8499
8499/M	8499/M	8533	8536	8544/M	8544/M	8544/M	8544/M
8608	8614	8615/M	8616/M	8630	8630	8630	8630/M
8630	8630/M	8630/M	8630	8630	8630	8630	8630
8630/M	8630	8630/M	8630	8630/M	8630	8630	8630
8630	8630	8630	8630	8630	8630/M	8630	8630/M
8630	8630/M	8630	8630	8630	8630	8630	8630
8630/M	8630	8630	8630/M	8630	8630/M	8630	8630
8630	8630	8630	8630	8630	8630	8630	8630/M
8630	8630/M	8630	8630/M	8630	8630	8630	8630
8630	8630	8630	8630	8630/M	8630	8630/M	8630/M
8635/M	8635	8635/M	8741/M	8741/M	8741/M	8741/M	8742
8742/M	8742	8742/M	8753	8754	8754	8754	8813/M
8834/M	8835/M	8837/P	8839/P	8842	8844	8850/M	8913/M
8914/M	8919	8921	8927/M	8929/M	8930/M	9018	9102
9102	9102	9134	9189	9192	9196/P	9255	9259
9260	9261	9262	9264	9298/M	9299	9385	9387
9388	9389	9390	9394/M	9399	9400	9401/M	9403
9406	9406	9406	9406/M	9406	9406/M	9406/M	9406
9406	9406	9406	9406	9406/M	9406	9406/M	9406
9406/M	9406	9406	9406	9406	9406	9406	9406
9406	9406/M	9406	9406/M	9406	9406/M	9406	9406
9406	9406	9406	9406	9406/M	9406	9406	9406/M
9406	9406/M	9406	9406	9406	9406	9406	9406
9406	9406	9406	9406/M	9406	9406/M	9406	9406/M
9406	9406	9406	9406	9406	9406	9406	9406
9406/M	9406	9406/M	9406/M	9409/M	9410	9416	9417
9417	9417/M	9417	9418/M	9420/M	9422/M	9426/M	9429
9471	9472	9473	9474/M	9477	9479	9479	9479
9479/M	9479	9479/M	9479/M	9479	9479	9479	9479
9479	9479/M	9479	9479/M	9479	9479/M	9479	9479
9479	9479	9479	9479	9479	9479	9479/M	9479
9479/M	9479	9479/M	9479	9479	9479	9479	9479
9479	9479/M	9479	9479	9479/M	9479	9479/M	9479
9479	9479	9479	9479	9479	9479	9479	9479
9479/M	9479	9479/M	9479	9479/M	9479	9479	9479
9479	9479	9479	9479	9479	9479/M	9479	9479/M
9479/M	9482/M	9483	9515	9520	9521	9522	9523
9526/M	9527	9528	9528	9528/M	9528	9529/M	9531/M
9533/M	9536	9549	9554	9555	9556	9557	9560/M
9561	9562	9562	9562/M	9562	9563/M	9565/M	9567/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9570	9612	9616	9617	9618/M	9621	9623	9623
		9623	9623/M	9623	9623/M	9623/M	9623	9623	9623
		9623	9623	9623/M	9623	9623/M	9623	9623/M	9623
		9623	9623	9623	9623	9623	9623	9623	9623/M
		9623	9623/M	9623	9623/M	9623	9623	9623	9623
		9623	9623	9623/M	9623	9623	9623/M	9623	9623/M
		9623	9623	9623	9623	9623	9623	9623	9623
		9623	9623/M	9623	9623	9623	9623/M	9623	9623
		9623	9623	9623	9623	9623	9623/M	9623	9623/M
		9623/M	9623/M	9626/M	9628	9636	9640	9641	9642/M
		9645	9647	9647	9647	9647/M	9647	9647/M	9647/M
		9647	9647	9647	9647	9647	9647/M	9647	9647/M
		9647	9647/M	9647	9647	9647	9647	9647	9647
		9647	9647	9647/M	9647	9647/M	9647	9647/M	9647
		9647	9647	9647	9647	9647	9647/M	9647	9647
		9647/M	9647	9647/M	9647	9647	9647	9647	9647
		9647	9647	9647	9647	9647/M	9647	9647/M	9647
		9647/M	9647	9647	9647	9647	9647	9647	9647
		9647	9647/M	9647	9647/M	9647/M	9650/M	9652	9883
		9883	9884	10080	10104	10106/P	10106/P	10111/M	10116/M
		10129	10131	10133/M	10134/M	10192	10198	10198	10198/M
		10198	10201	10201	10201	10201/M	10201	10201/M	10201/M
		10201	10201	10201	10201	10201	10201/M	10201	10201/M
		10201	10201/M	10201	10201	10201	10201	10201	10201
		10201	10201	10201/M	10201	10201/M	10201	10201/M	10201
		10201	10201	10201	10201	10201	10201/M	10201	10201
		10201/M	10201	10201/M	10201	10201	10201	10201	10201
		10201	10201	10201	10201	10201/M	10201	10201/M	10201
		10201/M	10201	10201	10201	10201	10201	10201	10201
		10201	10201/M	10201	10201/M	10201/M	10473	10512/P	10514/M
		10516	10519	10519	10519	10519/M	10519	10519/M	10519/M
		10519	10519	10519	10519	10519	10519/M	10519	10519/M
		10519	10519/M	10519	10519	10519	10519	10519	10519
		10519	10519	10519/M	10519	10519/M	10519	10519/M	10519
		10519	10519	10519	10519	10519	10519/M	10519	10519
		10519/M	10519	10519/M	10519	10519	10519	10519	10519
		10519	10519	10519	10519	10519/M	10519	10519/M	10519
		10519/M	10519	10519	10519	10519	10519	10519	10519
		10519	10519/M	10519	10519/M	10519/M	10530	10530	10530
		10550	10550	10550/M	10550	10550	10550	10550/M	10550
		10550/M	10550/M	10550	10550	10550	10550	10550	10550/M
		10550	10550/M	10550	10550/M	10550	10550	10550	10550
		10550	10550	10550	10550	10550/M	10550	10550/M	10550
		10550/M	10550	10550	10550	10550	10550	10550	10550/M
		10550	10550	10550/M	10550	10550/M	10550	10550	10550
		10550	10550	10550	10550	10550	10550	10550/M	10550
		10550/M	10550	10550/M	10550	10550	10550	10550	10550
		10550	10550	10550	10550/M	10550	10550/M	10550/M	10550
		10561	10561	10561	10561	10561/M	10561/M	10561/M	10561/M
		10579	10579	10579	10579/M	10579	10579/M	10579/M	10579
		10579	10579	10579	10579	10579/M	10579	10579/M	10579
		10579/M	10579	10579	10579	10579	10579	10579	10579
		10579	10579/M	10579	10579/M	10579	10579/M	10579	10579

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10579	10579	10579	10579	10579/M	10579	10579	10579/M
		10579	10579/M	10579	10579	10579	10579	10579	10579
		10579	10579	10579	10579/M	10579	10579/M	10579	10579/M
		10579	10579	10579	10579	10579	10579	10579	10579
		10579/M	10579	10579/M	10579/M	10633	10677	10704	10725
		10725	10725/M	10725	10750	10816	10848	10868	10869
		10870/M	10870	10896	10896	10899	10901/M	10925	10925
		10925	10925/M	10925	10925/M	10925/M	10925	10925	10925
		10925	10925	10925/M	10925	10925/M	10925	10925/M	10925
		10925	10925	10925	10925	10925	10925	10925	10925/M
		10925	10925/M	10925	10925/M	10925	10925	10925	10925
		10925	10925	10925/M	10925	10925	10925/M	10925	10925/M
		10925	10925	10925	10925	10925	10925	10925	10925
		10925	10925/M	10925	10925/M	10925	10925/M	10925	10925
		10925	10925	10925	10925	10925	10925	10925/M	10925
		10925/M	10925/M	10984	10997	10997	10997/M	10997	10998
		11015/M	11015	11016/M	11077	11078	11085/M	11086/M	11086/M
		11086/M	11086/M	11117	11124	11124	11124	11124	11124/M
		11124/M	11124/M	11124/M	11139	11139	11139	11139/M	11139
		11139/M	11139/M	11139	11139	11139	11139	11139	11139/M
		11139	11139/M	11139	11139/M	11139	11139	11139	11139
		11139	11139	11139	11139	11139/M	11139	11139/M	11139
		11139/M	11139	11139	11139	11139	11139	11139	11139/M
		11139	11139	11139/M	11139	11139/M	11139	11139	11139
		11139	11139	11139	11139	11139	11139	11139/M	11139
		11139/M	11139	11139/M	11139	11139	11139	11139	11139
		11139	11139	11139	11139/M	11139	11139/M	11139/M	11146/P
		11148/M	11150	11151	11151	11151	11151/M	11151	11151/M
		11151/M	11151	11151	11151	11151	11151	11151/M	11151
		11151/M	11151	11151/M	11151	11151	11151	11151	11151
		11151	11151	11151	11151/M	11151	11151/M	11151	11151/M
		11151	11151	11151	11151	11151	11151	11151/M	11151
		11151	11151/M	11151	11151/M	11151	11151	11151	11151
		11151	11151	11151	11151	11151	11151/M	11151	11151/M
		11151	11151/M	11151	11151	11151	11151	11151	11151
		11151	11151	11151/M	11151	11151/M	11151/M		
tmv$subsystem_prior_threshold	6365	6911	7863	7866	7898	7979	8172	8227	8499
		8630	9406	9479	9623	9647	10201	10486	10519
		10550	10579	10925	11139	11151			
tmv$swapin_in_progress	6340	10736							
tmv$tables_initialized	6349	8629	8634	8637/M	8899	8999	10443		
tmv$time_to_call_dispatcher	6341	10752/M	10814/M	10816/M					
tmv$timed_wait_not_queued	6342	9035	9115						
tmv$timed_wait_queue	6343	7877/P	8288	8288	8288/M	8288/M	8288	8288/M	8288
		8288/M	8319	8323	8324/M	8325/M	8338	8339/M	8344
		8345/M	10550/S	10550	10550	10550/M	10550/S	10550	10550
		10550/M	10561	10561	10561/M	10561/M	10561	10561/M	10561
		10561/M	10747	10750/S	10813	10816/S	10896/S	10896/S	10896
		10897	10898	10900/M	10932	10933/M	11124	11124	11124/M
		11124/M	11124	11124/M	11124	11124/M			
tmv$total_task_count	6344	8455/M	8455	8542/M	8542	8640/M	8640	8751/M	8751
		11039/M	11039	11087/M	11087				
tos	7565	7589/M	7591/M	7593/M	7595/M				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tos_registers	3150	7595							
total_cptime	8682	8739/M	8739	8739					
total_cptime	9680	9742/M	9744	9746					
total_cptime	10408	10466/M	10466	10466					
update_cp_statistics	9661	8739	9749	10466					
update_dispatching_controls	9753	8739	9694	9818	10466	10774			
update_priority_integer	9826	8739	8739	8739	9694	9694	9694	9768	9782
		9807	9843	10466	10466	10466	10774	10774	10774
update_scheduling_priority	10314	10321/M	10342						
update_timed_wait_queue	10879	10550	10936						
user_condition_register	3113	9021/M	9022	9155/M	9156	10791/M	10791	10849/M	10849
user_mask	3109	10845	10846/M	10846					
user_requested_dispatching_prio	1451	10017/M	10032/M	10327	10328	10333			
v1	2694	8447/M							
wait_inhibit	5537	9008	10451						
wait_inhibited	2521	9009/M	9142	9162	9179/M	10802/M			
wait_inhibited	4956	7922	7923	7924/M	9143	9163	9169/M	9171/M	9180/M
		10801	10804/M	10848					
x_registers	3140	8219							
xcb_offset	4933	6625/M	7054	7054	7056	7056	7058	7107	7150
		7219	7259	7321	7464	7510	7544	7582	7612
		7722	7810	7866	7866	7866	7866	7866	7898
		7898	7898	7898	7898	7978	7979	7979	7979
		7979	7979	8170	8172	8172	8172	8172	8172
		8218	8227	8227	8227	8227	8227	8445/M	8499
		8499	8499	8499	8499	8608/M	8630	8630	8630
		8630	8630	9255	9385	9406	9406	9406	9406
		9406	9471	9479	9479	9479	9479	9479	9515
		9549	9612	9623	9623	9623	9623	9623	9636
		9647	9647	9647	9647	9647	10201	10201	10201
		10201	10201	10516	10519	10519	10519	10519	10519
		10550	10550	10550	10550	10550	10550	10579	10579
		10579	10579	10579	10639	10682	10921	10925	10925
		10925	10925	10925	11002/M	11139	11139	11139	11139
		11139	11150	11151	11151	11151	11151	11151	
xcb_offset	10954	11002							
xcb_p	1157	8006/M	8006	8425	8493/P	8557	8558/M	8623	8650
		8675/P	8703	8708	8713/P	8723/P	8724	8733	8733
		8739/M	8739	8739/M	8739	8739	8739	8739/M	8739/M
		8739/M	8739/M	8743/M	8745/M	8900	8901	8902	8903/M
		8904	8905	8905	8907/M	8908/M	8913	8926/M	9006
		9007	9009/M	9021/M	9021	9023/M	9023	9024/M	9024
		9033	9041	9113	9121	9142	9155/M	9155	9157/M
		9157	9158/M	9158	9162	9168	9179/M	9188	9189/S
		9190	9192/S	9707/M	9707	9709/M	9709	9725	9726
		9728/M	9730/M	9733/M	9735/M	9921	9924	9924	9930
		9956	9956	9962/M	9962	9963/M	9964/M	10128/M	10129/M
		10129	10131/M	10131	10459	10466/M	10466	10466/M	10466
		10466	10466	10466/M	10466/M	10466/M	10466/M	10529/M	10570/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10745/M	10785/M	11105	11133/M				
xcb_p	4987	8426	8443						
xcb_p	5006	8605							
xcb_p	5567	5570	5571						
xcb_p	6568	6621/M	6622/M	6625	6626				
xcb_p	6864	6893	6895	6897/M	6900/M	6901/M	6909	6910	6911
		6912	6915	6917/M	6923/M	6923	6925/M	6928/M	6948
		7054/P	7056/P	7058/P					
xcb_p	7418	7463/M							
xcb_p	7474	7507/M	7509/M						
xcb_p	7519	7534/M	7541/M	7543/M					
xcb_p	7571	7581/M	7588	7590	7592	7593	7595		
xcb_p	7610	7612/M							
xcb_p	7623	7719/M	7721/M						
xcb_p	7766	7866	7866	7866/M	7866/M	7866/M	7866	7866	7866
		7866	7866	7866/M	7866/M	7866	7866/M	7866/M	7866
		7866/P	7866/P	7866/P	7898	7898	7898/M	7898/M	7898/M
		7898	7898	7898	7898	7898	7898/M	7898/M	7898
		7898/M	7898/M	7898	7898/P	7898/P	7898/P		
xcb_p	7802	7810/M	7837	7861	7862	7863	7866/P	7898/P	
xcb_p	7932	7979	7979	7979/M	7979/M	7979/M	7979	7979	7979
		7979	7979	7979/M	7979/M	7979	7979/M	7979/M	7979
		7979/P	7979/P	7979/P					
xcb_p	7952	7978/M	7979/P						
xcb_p	8110	8172	8172	8172/M	8172/M	8172/M	8172	8172	8172
		8172	8172	8172/M	8172/M	8172	8172/M	8172/M	8172
		8172/P	8172/P	8172/P					
xcb_p	8130	8170/M	8172/P						
xcb_p	8182	8221	8221						
xcb_p	8182	8227	8227	8227/M	8227/M	8227/M	8227	8227	8227
		8227	8227	8227/M	8227/M	8227	8227/M	8227/M	8227
		8227/P	8227/P	8227/P					
xcb_p	8214	8218/M	8219	8221/P	8227/P				
xcb_p	8381	8445/M	8445/M	8445	8445				
xcb_p	8381	8499	8499	8499/M	8499/M	8499/M	8499	8499	8499
		8499	8499	8499/M	8499/M	8499	8499/M	8499/M	8499
		8499/P	8499/P	8499/P					
xcb_p	8415	8443/M	8445/P	8454/M	8481/M	8484/M	8484	8486/M	8486
		8488/M	8493/P	8496/M	8499/P				
xcb_p	8529	8557/M	8567/P						
xcb_p	8581	8608/M	8608/M	8608	8608				
xcb_p	8581	8630	8630	8630/M	8630/M	8630/M	8630	8630	8630
		8630	8630	8630/M	8630/M	8630	8630/M	8630/M	8630
		8630/P	8630/P	8630/P					
xcb_p	8602	8606/M	8608/P	8622/M	8623/M	8624/M	8630/P	8646/M	8647/M
		8650/M	8655/M	8658	8661/M	8665/M	8669/M	8669	8671/M
		8671	8673/M	8675/P					
xcb_p	8762	8765/M	8765						
xcb_p	8773	8788	8792	8802/M	8802	8805	8806	8807	8813
		8814/M	8818/S	8820	8821	8821	8823/M	8824/M	8828/S
		8834	8837/S	8839/S	8849/M	8853	8854/M	8855/P	
xcb_p	8889	8917/P	8918	8918	8919/S	8921/S	8923/P		
xcb_p	9217	9255/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	9224	9255/P	9256						
xcb_p	9349	9406	9406	9406/M	9406/M	9406/M	9406	9406	9406
		9406	9406	9406/M	9406/M	9406	9406/M	9406/M	9406
		9406/P	9406/P	9406/P					
xcb_p	9363	9385/M	9386	9389	9406/P	9425			
xcb_p	9446	9479	9479	9479/M	9479/M	9479/M	9479	9479	9479
		9479	9479	9479/M	9479/M	9479	9479/M	9479/M	9479
		9479/P	9479/P	9479/P					
xcb_p	9454	9471/M	9479/P						
xcb_p	9497	9515/M	9516	9517	9518	9519	9522	9549/M	9550
		9551	9552	9553	9556				
xcb_p	9584	9623	9623	9623/M	9623/M	9623/M	9623	9623	9623
		9623	9623	9623/M	9623/M	9623	9623/M	9623/M	9623
		9623/P	9623/P	9623/P	9647	9647	9647/M	9647/M	9647/M
		9647	9647	9647	9647	9647	9647/M	9647/M	9647
		9647/M	9647/M	9647	9647/P	9647/P	9647/P		
xcb_p	9592	9612/M	9613	9614	9615	9623/P	9636/M	9637	9638
		9639	9647/P						
xcb_p	10157	10201	10201	10201/M	10201/M	10201/M	10201	10201	10201
		10201	10201	10201/M	10201/M	10201	10201/M	10201/M	10201
		10201/P	10201/P	10201/P					
xcb_p	10166	10171/P	10172	10176	10176	10177	10178	10180	10183
		10190/M	10191	10193/M	10196	10199	10201/P	10203	10207/P
xcb_p	10230	10234/P	10235	10239	10239	10240	10241/M	10243/P	
xcb_p	10408	10516/M							
xcb_p	10408	10519	10519	10519/M	10519/M	10519/M	10519	10519	10519
		10519	10519	10519/M	10519/M	10519	10519/M	10519/M	10519
		10519/P	10519/P	10519/P	10550	10550	10550/M	10550/M	10550/M
		10550	10550	10550	10550	10550	10550/M	10550/M	10550
		10550/M	10550/M	10550	10550/P	10550/P	10550/P	10579	10579
		10579/M	10579/M	10579/M	10579	10579	10579	10579	10579
		10579/M	10579/M	10579	10579/M	10579/M	10579	10579/P	10579/P
		10579/P							
xcb_p	10408	10550/M	10550/M	10550/M	10550/P				
xcb_p	10434	10459/M	10463	10469/M	10479	10482	10485/M	10486	10487/M
		10487	10490/M	10554	10579/P	10639/M	10642	10644	10645
		10649	10650	10682/M	10685	10687	10689	10690	10695
		10696	10785	10786	10791/M	10791	10793/M	10793	10797/M
		10797	10802/M	10809					
xcb_p	10835	10840	10840	10841/M	10842/M	10844/M	10845	10846/M	10846
		10847/M	10847	10849/M	10849				
xcb_p	10879	10925	10925	10925/M	10925/M	10925/M	10925	10925	10925
		10925	10925	10925/M	10925/M	10925	10925/M	10925/M	10925
		10925/P	10925/P	10925/P					
xcb_p	10889	10921/M	10922/M	10923/M	10925/P				
xcb_p	11095	11139	11139	11139/M	11139/M	11139/M	11139	11139	11139
		11139	11139	11139/M	11139/M	11139	11139/M	11139/M	11139
		11139/P	11139/P	11139/P	11151	11151	11151/M	11151/M	11151/M
		11151	11151	11151	11151	11151	11151/M	11151/M	11151
		11151/M	11151/M	11151	11151/P	11151/P	11151/P		
xcb_p	11095	11150/M							
xcb_p	11103	11105/M	11109	11110/M	11113/M	11139/P			
xcb_search	10160	10171/P							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_search	10227	10234/P							
xp	2509	7590	7592	7593	7595	8006/M	8006	8219	8484/M
		8484	8486/M	8486	8488/M	8669/M	8669	8671/M	8671
		8673/M	8733	8733	9007	9021/M	9021	9155/M	9155
		9168	9924	9924	9956	9956	9963/M	9964/M	10791/M
		10791	10840	10840	10841/M	10842/M	10844/M	10845	10846/M
		10846	10849/M	10849					

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

```
        2 MODULE tmm$mtr_flag_signal_functions;
        3
        4 (*
        5 (*  PURPOSE:
        6 (*     This module processes flag and signal handling job mode monitor requests
        7 (*     and monitor mode requests.
        8 (*
        9
```

```
o  1025 (*  Define arbitrary constant that is used to check if there is enough space in SFSA of current
o  1026 (*  task to terminate it.
o  1027
o  1028   CONST
o  1029     task_termination_stack_area = 30000;
o  1030
o  1031
o  1032 (*Define constants for recognizing hung tasks.
o  1033
o  1034   VAR
o  1035     tmv$halt_on_hung_task: [XDCL, #GATE] boolean := FALSE,
o  1036     tmv$system_debug_ring: [XDCL, #GATE] integer := 0,
o  1037     tmv$system_debug_segment: [XDCL, #GATE] integer := 0,
o  1038     tmv$job_debug_ring_p: [XDCL, #GATE] ^ost$ring := NIL,
o  1039     tmv$system_error_hang_count: [XDCL, #GATE] 0 .. 0ffffffff(16) := 6;
o  1040
o  1041 (*External procedures called by tmp$mtr_task_manager_functions.

o  1043
o  1044   PROCEDURE [XREF] dpp$display_error
o  1045     (     line: string ( * <= dpc$top_line_message_size));
o  1046
   1058
   1059   PROCEDURE [INLINE] jmp$unlock_ajl
   1060     (     ijle_p: ^jmt$initiated_job_list_entry);
   1061
   1062     VAR
   1063       ajlo: jmt$ajl_ordinal;
   1064
   1065     tmp$set_lock (tmv$ptl_lock);
   1066     ajlo := ijle_p^.ajl_ordinal;
   1067     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
   1068       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
   1069     ELSE
   1070       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
   1071     IFEND;
   1072     tmp$clear_lock (tmv$ptl_lock);
   1073
   1074   PROCEND jmp$unlock_ajl;
   1075
o  2239
o  2240   PROCEDURE [INLINE] jmp$unlock_ajl_with_lock
o  2241     (     ijle_p: ^jmt$initiated_job_list_entry);
o  2242
o  2243     VAR
```



```
o  2244       ajlo: jmt$ajl_ordinal;
o  2245
o  2246     ajlo := ijle_p^.ajl_ordinal;
o  2247     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
o  2248       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
o  2249     ELSE
o  2250       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
o  2251     IFEND;
o  2252
o  2253   PROCEND jmp$unlock_ajl_with_lock;
o  2254
   2258
   2259   PROCEDURE [XREF] tmp$delay (VAR rb: tmt$rb_delay;
   2260        cst_p: ^ost$cpu_state_table);
   2261
o  3587
o  3588   PROCEDURE [XREF] tmp$check_taskid (taskid: ost$global_task_id;
o  3589         option: tmt$option;
o  3590     VAR status: syt$monitor_status);
o  3591
   3641
   3642   PROCEDURE [XREF] tmp$check_taskid_with_lock_set
   3643     (     taskid: ost$global_task_id;
   3644           option: tmt$option;
   3645      VAR status: syt$monitor_status);
   3646
o  3649
o  3650   PROCEDURE [XREF] tmp$set_task_ready (task_id: ost$global_task_id;
o  3651         readying_task_priority: jmt$dispatching_priority;
o  3652         ready_condition: tmt$ready_condition);
o  3653
   3656
   3657   PROCEDURE [XREF] tmp$find_xcb (taskid: ost$global_task_id;
   3658     VAR xcb_p: ^ost$execution_control_block;
   3659     VAR ijle_p: ^jmt$initiated_job_list_entry;
   3660     VAR status: syt$monitor_status);
   3661
o  3666   PROCEDURE [XREF] tmp$get_xcb_p (task_id: ost$global_task_id;
o  3667     VAR xcb_p: ^ost$execution_control_block;
o  3668     VAR ijle_p: ^jmt$initiated_job_list_entry);
o  3669
   3672   PROCEDURE [XREF] mmp$fetch_stack_segment_info (xcb_p: ^ost$execution_control_block;
   3673         ring: ost$valid_ring;
   3674         set_length_to_zero: boolean;
   3675     VAR stack_segment_number: ost$segment;
   3676     VAR maximum_segment_length: ost$segment_length;
   3677     VAR found: boolean);
o  3680   PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
   3691
   3692
   3693 (*  PURPOSE:     procedure mtp$error_stop
   3694 (*    Prefixes 'ERR=VEOS1000-' to the string and calls mtp$step_unstep_system to write string and step system)
   3695
   3696   PROCEDURE [XREF] mtp$error_stop (text: string(*<=63) );
   3697
   3698   PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
   3699         condition: osc$max_status_condition_number + 1 .. 0fffffffff(16);
```

```
                3700      VAR status: syt$monitor_status);
                3701
              o 3711
              o 3712 {  PURPOSE:    procedure mtp$step_unstep_system
              o 3713 {     Writes a line of text to the Top Line of the console.  If the idle code specified indicates an error
              o 3714 {     (i.e. not an operator command or a recovery), then 'ERR=' must be first 4 characters or it will be
              o 3715 {     prefixed.  Then the system is Terminated, Aborted, Stepped, or Idled per the idle code specified.
              o 3716 {     All messages sent to the Top Line must have a VEOSxxxx code in the text.  Guidelines for using
              o 3717 {     VEOSxxxx codes and a list of all VEOSxxxx codes are documented in the comments found in the proc
              o 3718 {     dpp$display_error which is in the deck dpm$system_console_monitor.
              o 3719
              o 3720   PROCEDURE [XREF] mtp$step_unstep_system
              o 3721         (term_code: syt$180_idle_code; text: string(*<=76) );
              o 3722


                3726 {External variables.
                3727
                3728 {If a MCR fault occurs in a task executing in a ring <= the value
                3729 {of this variable, Monitor will halt the CPU. The variable is located
                3730 {in Monitor and is intended to be used for debug only.
                3731
                3732   VAR
                3733     mtv$system_haltring: [XREF] 0 .. 255,
                3734     mtv$halt_cpu_ring_number: [XREF] 0 .. 255;
                3735
                3736   VAR
                3737     tmv$ptl_p: [XREF] ^tmt$primary_task_list;
                3738
              o 3741 {Define SMU Communications Block (SCB).
              o 3742
              o 3743   VAR
              o 3744     mtv$scb: [XREF] mtt$smu_communications_block;
              o 3880
```



```
              o 3883
              o 3884   PROCEDURE [XDCL] tmp$set_monitor_flag
              4 3885     (     task_id (input) : ost$global_task_id;
              4 3886           flag_id (input) : syt$monitor_flag;
              4 3887      VAR status (output) : syt$monitor_status);
              4 3888
              4 3889
              4 3890     VAR
              4 3891       cst_p: ^ost$cpu_state_table;
              4 3892
              4 3893     #KEYPOINT (osk$debug, osk$m * task_id.index, tmk$set_monitor_flag);
             10 3894
             10 3895     status.normal := TRUE;
             10 3896     mtp$cst_p (cst_p);
             2A 3897
             2A 3898 {  If it is the current task, set the free flag in UCR.
             2A 3899
             2A 3900     tmp$set_lock (tmv$ptl_lock);
             70 3901
             70 3902     tmp$check_taskid_with_lock_set (task_id, tmc$opt_return, status);
             8C 3903     IF NOT status.normal THEN
             94 3904       tmp$clear_lock (tmv$ptl_lock);
             CC 3905       RETURN;
             CE 3906     IFEND;
             CE 3907
             CE 3908     IF task_id = cst_p^.taskid THEN
             DE 3909       cst_p^.xcb_p^.xp.user_condition_register := cst_p^.xcb_p^.xp.
             DE 3910             user_condition_register + $ost$user_conditions [osc$free_flag];
             DE 3911       cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags + $syt$monitor_flags [flag_id];
             DE 3912       IF tmv$ptl_p^ [task_id.index].new_task_status < tmc$ts_first_ready_uncond THEN
            124 3913         tmv$ptl_p^ [task_id.index].new_task_status := tmc$ts_null;
            128 3914       IFEND;
            128 3915       tmp$clear_lock (tmv$ptl_lock);
            15E 3916       RETURN;
            160 3917     IFEND;
            160 3918
            160 3919     IF flag_id IN tmv$ptl_p^ [task_id.index].monitor_flags THEN
            184 3920       tmp$clear_lock (tmv$ptl_lock);
            1BC 3921       RETURN;
            1BE 3922     IFEND;
            1BE 3923
            1BE 3924 { Set monitor flag in PTL.
            1BE 3925
            1BE 3926     tmv$ptl_p^ [task_id.index].monitor_flags := tmv$ptl_p^ [task_id.index].
            1BE 3927           monitor_flags + $syt$monitor_flags [flag_id];
            1BE 3928     tmp$set_task_ready (task_id, 0 {readying_task_priority}, tmc$rc_ready_conditional);
            200 3929     tmp$clear_lock (tmv$ptl_lock);
            236 3930
            236 3931   PROCEND tmp$set_monitor_flag;
```

```
0  3933
0  3934   PROCEDURE [XDCL] tmp$set_system_flag
0  3935     (     task_id [input] : ost$global_task_id;
0  3936           flag_id [input] : ost$system_flag;
0  3937      VAR status [output] : syt$monitor_status);
0  3938
0  3939 {
0  3940 {   The purpose of this request is to set a system flag in the XCB of the task
0  3941 { specified. System flags can be set in the XCB of any valid task including
0  3942 { tasks currently swapped out. If the specified task is swapped out, swapin
0  3943 { will be initiated by this request.
0  3944 {
0  3945 {         TMP$SET_SYSTEM_FLAG (TASKID, FLAGID, STATUS)
0  3946 {
0  3947 { TASKID: (input) This parameter specifies the taskid of the task to receive the
0  3948 {          flag.
0  3949 {
0  3950 { FLAGID: (input) This parameter specifies the flag id to be set.
0  3951 {
0  3952 { STATUS: (output) This parameter specifies the request status.
0  3953 {
0  3954 {
0  3955
0  3956     VAR
0  3957       cst_p: ^ost$cpu_state_table,
0  3958       ijle_p: ^jmt$initiated_job_list_entry,
0  3959       xcb_p: ^ost$execution_control_block;
0  3960
0  3961     #KEYPOINT (osk$debug, osk$m * task_id.index, tmk$set_system_flag);
10 3962
10 3963     status.normal := TRUE;
10 3964     mtp$cst_p (cst_p);
2A 3965
2A 3966 {  If it is the current task, set the free flag in UCR.
2A 3967
2A 3968     tmp$set_lock (tmv$ptl_lock);
70 3969
70 3970     tmp$check_taskid_with_lock_set (task_id, tmc$opt_return, status);
8C 3971     IF NOT status.normal THEN
94 3972       tmp$clear_lock (tmv$ptl_lock);
CC 3973       RETURN;
CE 3974     IFEND;
CE 3975
CE 3976     IF task_id = cst_p^.taskid THEN
DE 3977       IF cst_p^.xcb_p^.task_is_terminating THEN
EA 3978         mtp$set_status_abnormal ('TM', tme$invalid_global_taskid, status);
100 3979       ELSE
100 3980         cst_p^.xcb_p^.xp.user_condition_register := cst_p^.xcb_p^.xp.
100 3981               user_condition_register + $ost$user_conditions [osc$free_flag];
100 3982         cst_p^.xcb_p^.system_flags := cst_p^.xcb_p^.system_flags + $tmt$system_flags [flag_id];
100 3983         cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags +
100 3984               $syt$monitor_flags [tmc$mf_cause_job_free_flag_trap];
100 3985         cst_p^.xcb_p^.wait_inhibited := TRUE;
100 3986         IF tmv$ptl_p^ [task_id.index].new_task_status < tmc$ts_first_ready_uncond THEN
154 3987           tmv$ptl_p^ [task_id.index].new_task_status := tmc$ts_null;
158 3988         IFEND;
15A 3989       IFEND;
```



```
15A 3990       tmp$clear_lock (tmv$ptl_lock);
190 3991       RETURN;
192 3992     IFEND;
192 3993
192 3994     IF flag_id IN tmv$ptl_p^ [task_id.index].system_flags THEN
1B6 3995       tmp$clear_lock (tmv$ptl_lock);
1EE 3996       RETURN;
1F0 3997     IFEND;
1F0 3998
1F0 3999     tmp$get_xcb_p (task_id, xcb_p, ijle_p);
214 4000     IF (xcb_p <> NIL) AND xcb_p^.task_is_terminating THEN
22A 4001       status.normal := FALSE;
22A 4002       status.condition := tme$invalid_global_taskid;
22A 4003       jmp$unlock_ajl_with_lock (ijle_p);
282 4004       tmp$clear_lock (tmv$ptl_lock);
2B8 4005       RETURN;
2BA 4006     IFEND;
2BA 4007     IF (xcb_p <> NIL) THEN
2C4 4008       jmp$unlock_ajl_with_lock (ijle_p);
30A 4009     IFEND;
30A 4010
30A 4011 { Set system flag in PTL.
30A 4012
30A 4013     tmv$ptl_p^ [task_id.index].system_flags := tmv$ptl_p^ [task_id.index].
30A 4014           system_flags + $tmt$system_flags [flag_id];
30A 4015     IF (tmv$ptl_p^ [task_id.index].ptl_flags.wait_inhibited <> tmc$wi_wait_selected_r3) THEN
33E 4016       tmv$ptl_p^ [task_id.index].monitor_flags := tmv$ptl_p^ [task_id.index].
34C 4017             monitor_flags + $syt$monitor_flags [tmc$mf_cause_job_free_flag_trap];
34C 4018     IFEND;
34C 4019     tmp$set_task_ready (task_id, 0 [readying_task_priority], tmc$rc_ready_conditional_wi);
368 4020     tmp$clear_lock (tmv$ptl_lock);
39E 4021
39E 4022   PROCEND tmp$set_system_flag;
```

```
 0  4024
 0  4025   PROCEDURE [XDCL] tmp$flag_all_tasks
 0  4026     (     flag_id {input} : ost$system_flag;
 0  4027      VAR status {output} : syt$monitor_status);
 0  4028
 0  4029 {
 0  4030 {   The purpose of this request is to set a system flag in the XCB of every task
 0  4031 { in the system.  System flags can be set in the XCB of any valid task including
 0  4032 { tasks currently swapped out. If the specified task is swapped out, swapin
 0  4033 { will be initiated by this request.
 0  4034 {
 0  4035 {        TMP$FLAG_ALL_TASKS (FLAG_ID, STATUS)
 0  4036 {
 0  4037 { FLAG_ID: (input) This parameter specifies the flag id to be set.
 0  4038 {
 0  4039 { STATUS: (output) This parameter specifies the request status.
 0  4040 {
 0  4041
 0  4042     VAR
 0  4043       i: ost$task_index,
 0  4044       task_id: ost$global_task_id,
 0  4045       cst_p: ^ost$cpu_state_table;
 0  4046
 0  4047     status.normal := TRUE;
 4  4048
 4  4049     tmp$set_lock (tmv$ptl_lock);
46  4050     FOR i := 1 TO UPPERBOUND (tmv$ptl_p^) DO
5A  4051       IF tmv$ptl_p^ [i].status <> tmc$ts_null THEN
6A  4052
6A  4053         task_id.index := i;
6A  4054         task_id.seqno := tmv$ptl_p^ [i].sequence_number;
6A  4055
6A  4056 {  If it is the current task, set the free flag in UCR.
6A  4057
6A  4058         mtp$cst_p (cst_p);
8A  4059         IF task_id = cst_p^.taskid THEN
A8  4060           cst_p^.xcb_p^.xp.user_condition_register := cst_p^.xcb_p^.xp.
A8  4061                 user_condition_register + $ost$user_conditions [osc$free_flag];
A8  4062           cst_p^.xcb_p^.system_flags := cst_p^.xcb_p^.system_flags + $tmt$system_flags [flag_id];
A8  4063           cst_p^.xcb_p^.monitor_flags := cst_p^.xcb_p^.monitor_flags +
A8  4064                 $syt$monitor_flags [tmc$mf_cause_job_free_flag_trap];
A8  4065           cst_p^.xcb_p^.wait_inhibited := TRUE;
A8  4066           IF tmv$ptl_p^ [task_id.index].new_task_status < tmc$ts_first_ready_uncond THEN
F8  4067             tmv$ptl_p^ [task_id.index].new_task_status := tmc$ts_null;
FC  4068           IFEND;
FC  4069         IFEND;
FC  4070
FC  4071 { Set system flag in PTL.
FC  4072
FC  4073         tmv$ptl_p^ [task_id.index].system_flags := tmv$ptl_p^ [task_id.index].
FC  4074               system_flags + $tmt$system_flags [flag_id];
FC  4075         IF (tmv$ptl_p^ [task_id.index].ptl_flags.wait_inhibited <> tmc$wi_wait_selected_r3) THEN
130  4076           tmv$ptl_p^ [task_id.index].monitor_flags := tmv$ptl_p^ [task_id.index].
13E  4077                 monitor_flags + $syt$monitor_flags [tmc$mf_cause_job_free_flag_trap];
13E  4078         IFEND;
13E  4079         tmp$set_task_ready (task_id, 0 {readying_task_priority}, tmc$rc_ready_conditional_wi);
15A  4080       IFEND;
```



```
15A  4081     FOREND;
15E  4082     tmp$clear_lock (tmv$ptl_lock);
19A  4083
19A  4084   PROCEND tmp$flag_all_tasks;
```

```
 0  4086
 0  4087   PROCEDURE [XDCL] tmp$monitor_flag_job_tasks
 0  4088     (    monitor_flag_id: syt$monitor_flag;
 0  4089          ijle_p: ^jmt$initiated_job_list_entry);
 0  4090
 0  4091     VAR
 0  4092       taskid: ost$global_task_id,
 0  4093       status: syt$monitor_status;
 0  4094
 0  4095     tmp$set_lock (tmv$ptl_lock);
3E  4096     taskid := ijle_p^.job_monitor_taskid;
3E  4097
3E  4098     WHILE taskid.index <> 0 DO
4E  4099       taskid.seqno := tmv$ptl_p^ [taskid.index].sequence_number;
4E  4100       tmp$set_monitor_flag (taskid, monitor_flag_id, status);
86  4101       taskid.index := tmv$ptl_p^ [taskid.index].ijl_thread;
86  4102     WHILEND;
A0  4103
A0  4104     tmp$clear_lock (tmv$ptl_lock);
DC  4105   PROCEND tmp$monitor_flag_job_tasks;
```



```
 0  4107
 0  4108   PROCEDURE [XDCL] tmp$mtr_ready_task
 0  4109     (VAR rb [input, output] : tmt$rb_ready_task);
 0  4110
 0  4111
 0  4112 {
 0  4113 {  The purpose of this procedure is to process the job mode
 0  4114 {  request to change the status of a specified task to ready
 0  4115 {  and to set the WAIT INHIBITED flag in the task's PTL entry.
 0  4116 {
 0  4117 {    TMP$MTR_READY_TASK (RB)
 0  4118 {
 0  4119 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
 0  4120 {
 0  4121 {
 0  4122 {  The purpose of this request is to change the status of a specified task
 0  4123 {  to ready and to set the WAIT_INHIBITED flag in the task's PTL entry.
 0  4124 {
 0  4125 {  TYPE
 0  4126 {    TMT$RB_READY_TASK = RECORD
 0  4127 {      REQCODE,
 0  4128 {      STATUS,
 0  4129 {      TASK_ID,
 0  4130 {    RECEND;
 0  4131 {
 0  4132 {  REQCODE: (input) The value of this parameter is SYC$RC_READY_TASK.
 0  4133 {
 0  4134 {  STATUS: (output) This parameter specifies the standard monitor status.
 0  4135 {
 0  4136 {  TASK_ID: (input) This parameter specifies the global task id of the task to
 0  4137 {           ready.
 0  4138
 0  4139     VAR
 0  4140       readying_task_priority: jmt$dispatching_priority,
 0  4141       cst_p: ^ost$cpu_state_table;
 0  4142
 0  4143     tmp$check_taskid (rb.task_id, tmc$opt_return, rb.status);
2A  4144     IF rb.status.normal = FALSE THEN
32  4145       RETURN;
34  4146     IFEND;
34  4147
34  4148     mtp$cst_p (cst_p);
44  4149
44  4150     IF cst_p^.xcb_p^.dispatching_priority >=
76  4151          tmv$ptl_p^ [cst_p^.xcb_p^.global_task_id.index].readying_task_priority THEN
76  4152       readying_task_priority := cst_p^.xcb_p^.dispatching_priority;
7A  4153     ELSE
7A  4154       readying_task_priority := tmv$ptl_p^ [cst_p^.xcb_p^.global_task_id.index].readying_task_priority;
7C  4155     IFEND;
7C  4156
7C  4157     tmp$set_task_ready (rb.task_id, readying_task_priority, tmc$rc_ready_conditional_wi);
9A  4158
9A  4159   PROCEND tmp$mtr_ready_task;
```

```
 0  4161
 0  4162   PROCEDURE [XDCL] tmp$send_signal
 0  4163     (     task_id {input} : ost$global_task_id;
 0  4164           signal {input} : pmt$signal;
 0  4165      VAR status {output} : syt$monitor_status);
 0  4166
 0  4167
 0  4168 {
 0  4169 {   The purpose of this procedure is to place the signal in the
 0  4170 { specified task's XCB.  This procedure is callable only internal
 0  4171 { to monitor.
 0  4172 {
 0  4173 {       TMP$SEND_SIGNAL (TASKID, SIGNAL, STATUS)
 0  4174 {
 0  4175 { TASKID: (input) This parameter specifies the task in whose XCB
 0  4176 {       the signal is to be placed.
 0  4177 {
 0  4178 { SIGNAL: (input) This parameter specifies the signal block.
 0  4179 {
 0  4180 { STATUS: (output) This parameter is the standard monitor status.
 0  4181 {
 0  4182
 0  4183     VAR
 0  4184       cst_p: ^ost$cpu_state_table,
 0  4185       i (signal array index) : 1 .. tmc$maximum_signals,
 0  4186       ijle_p: ^jmt$initiated_job_list_entry,
 0  4187       xcb_p: ^ost$execution_control_block;
 0  4188
 0  4189     #KEYPOINT (osk$debug, osk$m * task_id.index, tmk$send_signal);
10  4190
10  4191     tmp$find_xcb (task_id, xcb_p, ijle_p, status);
3C  4192     IF status.normal = FALSE THEN
44  4193       RETURN;
46  4194     IFEND;
46  4195
46  4196   /access_xcb/
46  4197     BEGIN
46  4198       IF xcb_p^.task_is_terminating THEN
52  4199         status.normal := FALSE;
52  4200         status.condition := tme$invalid_global_taskid;
52  4201         EXIT /access_xcb/;
68  4202       IFEND;
68  4203
68  4204       IF xcb_p^.system_error_count >= tmv$system_error_hang_count THEN
78  4205         status.normal := FALSE;
78  4206         status.condition := pme$hung_recipient_task;
78  4207         EXIT /access_xcb/;
8E  4208       IFEND;
8E  4209
8E  4210     /find_free_signal_buffer/
8E  4211       BEGIN
8E  4212
8E  4213       /free_buffer_loop/
8E  4214         FOR i := 1 TO tmc$maximum_signals DO
98  4215           IF xcb_p^.signals.reserved [i] = FALSE THEN
AC  4216             EXIT /find_free_signal_buffer/;
B0  4217           IFEND;
```



```
B0  4218         FOREND /free_buffer_loop/;
B6  4219         #KEYPOINT (osk$unusual, 0, tmk$signal_buffers_full);
BA  4220         status.normal := FALSE;
BA  4221         status.condition := tme$mtr_signal_buffers_full;
BA  4222         EXIT /access_xcb/;
D2  4223       END /find_free_signal_buffer/;
D2  4224
D2  4225 {  Place signal in free signal buffer.
D2  4226
D2  4227       xcb_p^.signals.reserved [i] := TRUE;
D2  4228       xcb_p^.signals.present [i] := TRUE;
D2  4229       mtp$cst_p (cst_p);
F2  4230       xcb_p^.signals.buffer [i].originator := cst_p^.taskid;
F2  4231       xcb_p^.signals.buffer [i].signal := signal;
128 4232
128 4233       IF tmv$ptl_p^ [task_id.index].ptl_flags.wait_inhibited <> tmc$wi_wait_selected_r3 THEN
146 4234
146 4235 {  Set task status to ready and set free flag in specified tasks user condition
146 4236 {  register to invoke trap handler when the task gets the CPU.
146 4237
146 4238         xcb_p^.wait_inhibited := TRUE;
146 4239         tmp$set_monitor_flag (task_id, tmc$mf_cause_job_free_flag_trap, status);
16C 4240       ELSE
16C 4241         tmp$set_task_ready (task_id, 0 {readying_task_priority}, tmc$rc_ready_conditional);
18A 4242       IFEND;
18A 4243
18A 4244     END /access_xcb/;
18C 4245
18C 4246     jmp$unlock_ajl (ijle_p);
242 4247
242 4248   PROCEND tmp$send_signal;
```

```
 0  4250
 0  4251   PROCEDURE [XDCL] tmp$mtr_set_system_flag
 4  4252     (VAR rb [input, output] : tmt$rb_set_system_flag);
 4  4253
 4  4254
 4  4255 {
 4  4256 {  The purpose of this procedure is to process the job mode
 4  4257 {  request to set the specified system flag in the specified
 4  4258 {  task's PTL entry.
 4  4259 {
 4  4260 {    TMP$MTR_SET_SYSTEM_FLAG (RB)
 4  4261 {
 4  4262 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
 4  4263 {
 4  4264 {  The purpose of this request is to set a system flag in the XCB
 4  4265 {  (execution control block) of the specified task.
 4  4266 {
 4  4267 {  TYPE
 4  4268 {    TMT$RB_SET_SYSTEM_FLAG = RECORD
 4  4269 {      REQCODE,
 4  4270 {      STATUS,
 4  4271 {      TASK_ID,
 4  4272 {      FLAG_ID,
 4  4273 {    recend;
 4  4274 {
 4  4275 {  reqcode: (input) The value of this parameter is SYC$RC_MTR_SET_SYSTEM_FLAG.
 4  4276 {
 4  4277 {  task_id: (input) This parameter is the global task id of the task in
 4  4278 {           which the system flag is set.
 4  4279 {
 4  4280 {  flag_id: (input) This parameter specifies which system flag is to be set.
 4  4281 {
 4  4282 {  status: (output) This parameter is the standard monitor status.
 4  4283
 4  4284     tmp$set_system_flag (rb.task_id, rb.flag_id, rb.status);
2C  4285
2C  4286   PROCEND tmp$mtr_set_system_flag;
```



```
 0  4288
 0  4289   PROCEDURE [XDCL] tmp$mtr_send_signal
 4  4290     (VAR rb [input,output] : tmt$rb_send_signal);
 4  4291
 4  4292
 4  4293 {
 4  4294 {  The purpose of this procedure is to process the job mode
 4  4295 {  request to send signals to a specified task's XCB.
 4  4296 {
 4  4297 {    TMP$MTR_SEND_SIGNAL (RB)
 4  4298 {
 4  4299 {  RB : (INPUT,OUTPUT) This parameter specifies the request block.
 4  4300 {
 4  4301 {  TYPE
 4  4302 {    TMT$RB_SEND_SIGNAL = RECORD
 4  4303 {      REQCODE,
 4  4304 {      STATUS,
 4  4305 {      TASK_ID,
 4  4306 {      SIGNAL,
 4  4307 {    RECEND;
 4  4308 {
 4  4309 {  REQCODE: (input) The value of this parameter is SYC$RC_SEND_SIGNAL.
 4  4310 {  STATUS: (output) This parameter specifies standard system status.
 4  4311 {  TASK_ID: (input) This parameter specifies the task in whose XCB the
 4  4312 {           signal is to be placed.
 4  4313 {  SIGNAL: (input) This parameter specifies the signal block.
 4  4314 {
 4  4315
 4  4316     tmp$send_signal (rb.task_id, rb.signal, rb.status);
36  4317
36  4318   PROCEND tmp$mtr_send_signal;
 0  4319
```

```
 0  4321
 0  4322   PROCEDURE [XDCL] tmp$process_task_mcr_fault;
 0  4323
 0  4324 {       Purpose:
 0  4325 {         This procedure is called by the monitor interrupt processor to process
 0  4326 {         an MCR fault from a task if the MCR fault was selected by the task
 0  4327 {         to be processed in job mode.
 0  4328
 0  4329     VAR
 0  4330       cst_p: ^ost$cpu_state_table,
 0  4331       fault: ost$monitor_fault,
 0  4332       mcr_fault_p: ^tmt$mcr_faults,
 0  4333       xcb_p: ^ost$execution_control_block,
 0  4334       zero_pva: [STATIC] ost$pva := [0, 0, 0];
 0  4335
 0  4336
 0  4337 {Copy the fault information for the task's XCB to the signal record.
 0  4338
 0  4339     mtp$cst_p (cst_p);
14  4340     xcb_p := cst_p^.xcb_p;
14  4341     #KEYPOINT (osk$debug, xcb_p^.global_task_id.index * osk$m, tmk$process_task_mcr_fault);
38  4342     fault.identifier := tmc$mcr_fault;
38  4343     mcr_fault_p := #LOC (fault.contents);
42  4344     mcr_fault_p^.faults := xcb_p^.xp.monitor_condition_register;
42  4345     mcr_fault_p^.untranslatable_pointer := xcb_p^.xp.untranslatable_pointer;
42  4346
42  4347
42  4348 {Send the Monitor Fault to the task.
42  4349
42  4350     send_monitor_fault (xcb_p, ^fault, 'Job mode MCR fault', TRUE);
7A  4351
7A  4352
7A  4353 {Reset the XCB. Clear the UTP and MCR.
7A  4354
7A  4355     xcb_p^.xp.untranslatable_pointer := zero_pva;
7A  4356
7A  4357   PROCEND tmp$process_task_mcr_fault;
 0  4358
```



```
 0  4360
 0  4361   PROCEDURE [XDCL] tmp$process_unknown_req_fault;
 0  4362
 0  4363 {
 0  4364 {  This procedure is called by the monitor to process an unknown
 0  4365 {  system request fault issued by the current task.
 0  4366 {
 0  4367 {    TMP$PROCESS_UNKNOWN_REQ_FAULT
 0  4368 {
 0  4369
 0  4370     VAR
 0  4371       cst_p: ^ost$cpu_state_table,
 0  4372       fault: ost$monitor_fault,
 0  4373       xcb_p: ^ost$execution_control_block;
 0  4374
 0  4375 {  Set up the fault information in the signal block.
 0  4376
 0  4377     mtp$cst_p (cst_p);
14  4378     xcb_p := cst_p^.xcb_p;
14  4379     fault.identifier := tmc$unknown_system_req_fault;
14  4380
14  4381 {  Send the monitor fault to the task.
14  4382
14  4383     send_monitor_fault (xcb_p, ^fault, 'invalid monitor request', TRUE);
5A  4384
5A  4385   PROCEND tmp$process_unknown_req_fault;
 0  4386
```

```
  0  4388 {
  0  4389 {PROCEDURE tmp$send_monitor_fault;
  0  4390 {  PURPOSE:
  0  4391 {    The purpose of this procedure is to place a monitor fault in the
  0  4392 {    monitor fault buffer of the specified task.
  0  4393 {  NOTE:
  0  4394 {    The first monitor fault buffer is reserved for 'broken_task_monitor_fault'.
  0  4395 {    this procedure will start with the second monitor fault buffer when
  0  4396 {    searching for a free buffer.
  0  4397 {
  0  4398 ??FMT(FORMAT:=ON)??
  0  4399
  0  4400   PROCEDURE [XDCL] tmp$send_monitor_fault
  0  4401     (     task_id (input) : ost$global_task_id;
  0  4402           monitor_fault_p (input) : ^ost$monitor_fault;
  0  4403           check_traps_enabled (input) : boolean);
  0  4404
  0  4405     VAR
  0  4406       ijle_p: ^jmt$initiated_job_list_entry,
  0  4407       xcb_p: ^ost$execution_control_block;
  0  4408
  0  4409     tmp$get_xcb_p (task_id, xcb_p, ijle_p);
 28  4410     send_monitor_fault (xcb_p, monitor_fault_p, 'monitor fault', check_traps_enabled);
 52  4411     jmp$unlock_ajl (ijle_p);
10A  4412
10A  4413   PROCEND tmp$send_monitor_fault;
```



```
  0  4415
  0  4416   PROCEDURE [XDCL] tmp$mtr_process_system_error
  0  4417     (     rb: ost$rb_system_error);
  0  4418
  0  4419 {
  0  4420 {  This request is used by OSP$SYSTEM_ERROR to tell monitor that a
  0  4421 {  fatal system error occurred.
  0  4422 {
  0  4423 {  TYPE
  0  4424 {     OST$RB_SYSTEM_ERROR = RECORD
  0  4425 {       REQCODE,
  0  4426 {       FATAL,
  0  4427 {       STATUS,
  0  4428 {       CALLER_P_REGISTER,
  0  4429 {       STATUS_P,
  0  4430 {       TEXT_P,
  0  4431 {       CONDITION,
  0  4432 {       TEXT,
  0  4433 {     RECEND;
  0  4434 {
  0  4435 {  REQCODE: (input) The value of this parameter is SYC$RC_SYSTEM_ERROR.
  0  4436 {
  0  4437 {  FATAL: (input) This parameter specifies whether or not to halt the
  0  4438 {       system.
  0  4439 {
  0  4440 {  STATUS: (output) This specifies standard monitor status.
  0  4441 {
  0  4442 {  CALLER_P_REGISTER: (input) This parameter specifies the P register
  0  4443 {       where the problem occurred.
  0  4444 {
  0  4445 {  STATUS_P: (input)  This parameter specifies a pointer to a status
  0  4446 {       variable of possible importance to the error.
  0  4447 {
  0  4448 {  TEXT_P: (input) This parameter specifies a pointer to a text message
  0  4449 {       related to the failure.
  0  4450 {
  0  4451 {  CONDITION: (input) This parameter specifies the condition code found
  0  4452 {       in the status variable.
  0  4453 {
  0  4454 {  TEXT: (input) This parameter specifies the actual error message related
  0  4455 {       to the failure.
  0  4456
  0  4457     VAR
  0  4458       cst_p: ^ost$cpu_state_table,
  0  4459       fault: ost$monitor_fault,
  0  4460       error_message: string (80),
  0  4461       broken_task_fault_p: ^tmt$broken_task_monitor_fault;
  0  4462
  0  4463     mtp$cst_p (cst_p);
 14  4464
 14  4465     IF NOT mtv$sys_core_init_complete OR (cst_p^.xcb_p^.xp.p_register.pva.ring = 1) AND rb.fatal THEN
 50  4466       error_message (1,10) := 'VEOS1100- ';
 5E  4467       error_message (11,*) := rb.text;
 6C  4468       mtp$step_unstep_system (syc$ic_fatal_software_error, error_message(1,72));
 8E  4469     IFEND;
 8E  4470
 8E  4471     fault.identifier := tmc$broken_task_fault_id;
```

```
 8E  4472      broken_task_fault_p := #LOC (fault.contents);
 96  4473      broken_task_fault_p^.broken_task_condition := tmc$btc_system_error;
 96  4474      broken_task_fault_p^.trap_enable := cst_p^.xcb_p^.xp.trap_enable;
 96  4475      broken_task_fault_p^.status_p := rb.status_p;
 96  4476      broken_task_fault_p^.text_p := rb.text_p;
 96  4477      broken_task_fault_p^.caller_p_register := rb.caller_p_register;
 96  4478      error_message (1, *) := rb.text;
 D6  4479
 D6  4480      send_monitor_fault (cst_p^.xcb_p, ^fault, error_message, TRUE);
100  4481
100  4482
100  4483    PROCEND tmp$mtr_process_system_error;
```



```
  0  4485
  0  4486    PROCEDURE send_monitor_fault
  0  4487      (     xcb_p (input) : ^ost$execution_control_block;
  0  4488            monitor_fault_p (input) : ^ost$monitor_fault;
  0  4489            mtr_flt_message (input) : string ( * ),
  0  4490            check_traps_enabled (input) : boolean);
  0  4491
  0  4492  {
  0  4493  {   PURPOSE:
  0  4494  {     The purpose of this procedure is to place the monitor fault into free
  0  4495  {     monitor fault buffer of specified task.  The free flag is set to preempt
  0  4496  {     specified task execution and process the monitor fault next time task executes.
  0  4497  {
  0  4498  {   NOTE:
  0  4499  {     The first monitor fault buffer is reserved for sending monitor fault to
  0  4500  {     task to inform it that it is considered a broken task.  This is to ensure
  0  4501  {     that a buffer full condition will never occur when a task is broken.
  0  4502  {     It is assumed the specified task is in ready status.
  0  4503  {
  0  4504  {     If broken task processing aborts (in job mode), we will hang the task
  0  4505  {     unless it is a critical task or has system tables locked, in which case
  0  4506  {     we will halt the system.  We will not recurse back through here.
  0  4507  {
  0  4508  {     If a task is broken tmv$system_error_hang_count different times it
  0  4509  {     will be considered a hung task.  It will be processed as if broken
  0  4510  {     task processing had aborted.  (See above.)
  0  4511  {
  0  4512
  0  4513      VAR
  0  4514        cst_p: ^ost$cpu_state_table,
  0  4515        i: 1 .. tmc$maximum_monitor_faults + 1,
  0  4516        fault_contents_p: ^tmt$broken_task_monitor_fault,
  0  4517        broken_task: boolean,
  0  4518        fault: ost$monitor_fault,
  0  4519        halt_message: string (72),
  0  4520        jdr_p: ^ost$ring,
  0  4521        status: syt$monitor_status;
  0  4522
  0  4523      #KEYPOINT (osk$debug, xcb_p^.global_task_id.index * osk$m, tmk$send_monitor_fault);
 10  4524      mtp$cst_p (cst_p);
 20  4525
 20  4526      IF NOT mtv$sys_core_init_complete OR (cst_p^.ajlo = 0) AND
 58  4527           (cst_p^.taskid = cst_p^.ijle_p^.job_monitor_taskid) THEN
 58  4528        halt_message (1,10) := 'VEOS1100- ';
 66  4529        halt_message (11,*) := mtr_flt_message;
 84  4530        mtp$step_unstep_system (syc$ic_fatal_software_error, halt_message(1,72));
 A6  4531      IFEND;
 A6  4532
 A6  4533      IF ((xcb_p^.xp.p_register.pva.ring <= mtv$halt_cpu_ring_number) OR
 D6  4534          (xcb_p^.xp.p_register.pva.ring <= mtv$system_haltring) AND (cst_p^.ajlo = 0))
 D6  4535            AND (check_traps_enabled) THEN
 D6  4536        halt_message (1,10) := 'VEOS9920- ';
 E4  4537        halt_message (11, * ) := mtr_flt_message;
102  4538        dpp$display_error('Software Err below Halt Ring, initiating Software Breakpoint');
11E  4539        mtv$scb.nos_180_status.system_status.step_status_block.requested_status := mtc$stepped_system;
11E  4540        mtp$step_unstep_system (syc$ic_software_breakpoint, halt_message);
148  4541      IFEND;
```

```
148  4542
148  4543 {  Set up the broken task fault.
148  4544
148  4545      fault.identifier := tmc$broken_task_fault_id;
148  4546      fault.pva := xcb_p^.xp.p_register.pva;
148  4547      fault.a0 := xcb_p^.xp.a0_dynamic_space_pointer;
148  4548      fault.a1 := xcb_p^.xp.a1_current_stack_frame;
148  4549      fault.a2 := xcb_p^.xp.a2_previous_save_area;
148  4550      fault_contents_p := #LOC (fault.contents);
170  4551      fault_contents_p^.p := xcb_p^.xp.p_register;
170  4552      fault_contents_p^.a0 := xcb_p^.xp.a0_dynamic_space_pointer;
170  4553      fault_contents_p^.trap_enable := xcb_p^.xp.trap_enable;
170  4554      fault_contents_p^.monitor_condition_register := xcb_p^.xp.monitor_condition_register;
170  4555      fault_contents_p^.user_condition_register := xcb_p^.xp.user_condition_register;
170  4556      fault_contents_p^.monitor_fault_id := monitor_fault_p^.identifier;
170  4557
170  4558      check_repair_trap_mechanism (xcb_p, check_traps_enabled, broken_task, fault_contents_p^.
1C2  4559            broken_task_condition);
1C2  4560
1C2  4561      i := 2;
1C2  4562      WHILE (i <= tmc$maximum_monitor_faults) AND (xcb_p^.monitor_faults.present [i]) DO
1DC  4563        i := i + 1;
1DC  4564      WHILEND;
1F4  4565
1F4  4566      IF i <= tmc$maximum_monitor_faults THEN
1FA  4567        xcb_p^.monitor_faults.buffer [i] := monitor_fault_p^;
20A  4568        xcb_p^.monitor_faults.buffer [i].pva := xcb_p^.xp.p_register.pva;
20A  4569        xcb_p^.monitor_faults.buffer [i].a0 := xcb_p^.xp.a0_dynamic_space_pointer;
20A  4570        xcb_p^.monitor_faults.buffer [i].a1 := xcb_p^.xp.a1_current_stack_frame;
20A  4571        xcb_p^.monitor_faults.buffer [i].a2 := xcb_p^.xp.a2_previous_save_area;
20A  4572        xcb_p^.monitor_faults.present [i] := TRUE;
236  4573      ELSE
236  4574        IF broken_task = FALSE THEN
23E  4575          broken_task := TRUE;
23E  4576          fault_contents_p^.broken_task_condition := tmc$btc_mntr_fault_buffer_full;
23E  4577          xcb_p^.xp.trap_enable := osc$traps_enabled;
254  4578        IFEND;
256  4579      IFEND;
256  4580
256  4581      IF broken_task OR (monitor_fault_p^.identifier = tmc$broken_task_fault_id) THEN
266  4582
266  4583        xcb_p^.system_error_count := xcb_p^.system_error_count + 1;
266  4584
266  4585        IF xcb_p^.system_error_count > (tmv$system_error_hang_count + 4) THEN
280  4586          dpp$display_error('Broken Task, System Error Count exceeds limit; Terminating System');
2A0  4587          halt_message (1,10) := 'VEOS2020- ';
2AA  4588          halt_message (11,*) := mtr_flt_message;
2C8  4589          mtp$step_unstep_system (syc$ic_fatal_software_error,halt_message);
2EE  4590
2EE  4591        ELSEIF (xcb_p^.system_error_count = tmv$system_error_hang_count) OR
2FC  4592              (xcb_p^.monitor_faults.present [1]) THEN
2FC  4593
2FC  4594 {  HUNG TASK
2FC  4595
2FC  4596          IF (xcb_p^.system_table_lock_count >= 256) OR (xcb_p^.critical_task) THEN
310  4597            halt_message (1,10) := 'VEOS2010- ';
31E  4598            halt_message (11,*) := mtr_flt_message;
```



```
33C  4599            mtp$step_unstep_system (syc$ic_fatal_software_error,halt_message);
362  4600          ELSE
362  4601            IF tmv$halt_on_hung_task THEN
36A  4602              mtv$scb.nos_180_status.system_status.step_status_block.requested_status := mtc$stepped_system;
36A  4603              halt_message (1,10) := 'VEOS9910- ';
380  4604              halt_message (11,*) := mtr_flt_message;
39E  4605              dpp$display_error ('Task hung, initiating software breakpoint');
3BA  4606              mtp$step_unstep_system (syc$ic_software_breakpoint,halt_message);
3DC  4607            IFEND;
3DC  4608            tmp$set_monitor_flag (xcb_p^.global_task_id, syc$mf_hang_task, status);
3FC  4609            cst_p^.ijle_p^.hung_task_in_job := TRUE;
408  4610          IFEND;
40A  4611
40A  4612        IFEND;
40A  4613      IFEND;
40A  4614
40A  4615      IF broken_task AND (xcb_p^.monitor_faults.present [1] = FALSE) THEN
41C  4616        xcb_p^.monitor_faults.buffer [1] := fault;
426  4617        xcb_p^.monitor_faults.present [1] := TRUE;
42A  4618      IFEND;
42A  4619
42A  4620      tmp$set_monitor_flag (xcb_p^.global_task_id, tmc$mf_cause_job_free_flag_trap, status);
44C  4621      IF xcb_p^.xp.p_register.pva.ring <= tmv$system_debug_ring THEN
460  4622        IF (tmv$system_debug_segment = 0) OR (xcb_p^.xp.p_register.pva.seg <= tmv$system_debug_segment) THEN
474  4623          tmp$set_monitor_flag (xcb_p^.global_task_id, syc$mf_invoke_sysdebug, status);
496  4624        IFEND;
498  4625      ELSE
498  4626        IF tmv$job_debug_ring_p <> NIL THEN
4A6  4627          jdr_p := #address (1, #segment (xcb_p), #offset (tmv$job_debug_ring_p));
4A6  4628          IF xcb_p^.xp.p_register.pva.ring <= jdr_p^ THEN
4C2  4629            tmp$set_monitor_flag (xcb_p^.global_task_id, syc$mf_invoke_sysdebug, status);
4E0  4630          IFEND;
4E0  4631        IFEND;
4E0  4632      IFEND;
4E0  4633
4E0  4634    PROCEND send_monitor_fault;
```

```
      0  4636
      0  4637   PROCEDURE check_repair_trap_mechanism
      0  4638     (     xcb_p: ^ost$execution_control_block;
      0  4639           check_traps_enabled: boolean;
      0  4640      VAR broken: boolean;
      0  4641      VAR fault_id: tmt$broken_task_condition);
      0  4642
      0  4643     VAR
      0  4644       found: boolean,
      0  4645       stack_segnum: ost$segment,
      0  4646       stack_length: ost$segment_length,
      0  4647       status: syt$monitor_status;
      0  4648
      0  4649     broken := FALSE;
      4  4650
      4  4651
      4  4652 {  Make sure that traps are enabled.
      4  4653
      4  4654     IF (check_traps_enabled = TRUE) AND (xcb_p^.xp.trap_enable <> osc$traps_enabled) THEN
     1E  4655       broken := TRUE;
     1E  4656       fault_id := tmc$btc_mf_traps_disabled;
     1E  4657       xcb_p^.xp.trap_enable := osc$traps_enabled;
     2E  4658     IFEND;
     2E  4659
     2E  4660
     2E  4661 {Validate A0.
     2E  4662
     2E  4663     mmp$fetch_stack_segment_info (xcb_p, xcb_p^.xp.p_register.pva.ring, { set_length_to_zero } FALSE,
     6E  4664           stack_segnum, stack_length, found);
     6E  4665     IF NOT found THEN
     6E  4666       mtp$error_stop ('BTC - lost the stack segment');
     8E  4667     IFEND;
     8E  4668     IF (#RING (xcb_p^.xp.a0_dynamic_space_pointer) <> xcb_p^.xp.p_register.pva.ring) OR
     C2  4669         (#SEGMENT (xcb_p^.xp.a0_dynamic_space_pointer) <> stack_segnum) OR
     C2  4670         (#OFFSET (xcb_p^.xp.a0_dynamic_space_pointer) < 0) OR
     C2  4671         (#OFFSET (xcb_p^.xp.a0_dynamic_space_pointer) + 37 * 8 > stack_length) THEN
     C2  4672       broken := TRUE;
     C2  4673       fault_id := tmc$btc_invalid_a0;
     C2  4674       xcb_p^.xp.a0_dynamic_space_pointer := #ADDRESS (xcb_p^.xp.p_register.pva.ring, stack_segnum,
     C2  4675           mmc$ring_crossing_offset);
     C2  4676       xcb_p^.xp.a2_previous_save_area := NIL;
     C2  4677       mmp$fetch_stack_segment_info (xcb_p, xcb_p^.xp.p_register.pva.ring, { set_length_to_zero } TRUE,
    126  4678           stack_segnum, stack_length, found);
    126  4679     IFEND;
    126  4680
    126  4681
    126  4682   PROCEND check_repair_trap_mechanism;
      0  4683 MODEND tmm$mtr_flag_signal_functions;

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

**** NO DIAGNOSTICS
```


IDENTIFIER	DEFINED ON LINE	REFERENCES							
a0	622	4552/M							
a0	689	4547/M	4569/M						
a0_dynamic_space_pointer	3192	4547	4552	4569	4668	4669	4670	4671	4674/M
a1	690	4548/M	4570/M						
a1_current_stack_frame	3196	4548	4570						
a2	691	4549/M	4571/M						
a2_previous_save_area	3198	4549	4571	4676/M					
access_xcb	4196	4196	4201	4207	4222	4244			
ajl_ordinal	1103	1066	2246	4003	4008	4246	4411		
ajlo	1063	1066/M	1067/S	1070/S	1070/S				
ajlo	2244	2246/M	2247/S	2250/S	2250/S				
ajlo	2296	4526	4534						
ajlo	3934	4003/M	4003/S	4003/S	4003/S	4008/M	4008/S	4008/S	4008/S
ajlo	4162	4246/M	4246/S	4246/S	4246/S				
ajlo	4400	4411/M	4411/S	4411/S	4411/S				
b	1059	1065	1065						
b	2187	2194	2195						
b	3884	3900	3900						
b	3934	3968	3968						
b	4025	4049	4049						
b	4087	4095	4095						
b	4162	4246	4246						
b	4400	4411	4411						
bc	1059	1065/M	1065	1065					
bc	2188	2191/M	2192	2196					
bc	3884	3900/M	3900	3900					
bc	3934	3968/M	3968	3968					
bc	4025	4049/M	4049	4049					
bc	4087	4095/M	4095	4095					
bc	4162	4246/M	4246	4246					
bc	4400	4411/M	4411	4411					
broken	4640	4649/M	4655/M	4672/M					
broken_task	4517	4558/P	4574	4575/M	4581	4615			
broken_task_condition	613	4473/M	4559/P	4576/M					
broken_task_fault_p	4461	4472/M	4473/M	4474/M	4475/M	4476/M	4477/M		
buffer	765	4567/M	4568/M	4569/M	4570/M	4571/M	4616/M		
buffer	3399	4230/M	4231/M						
caller_p_register	43	4477							
caller_p_register	615	4477/M							
check_repair_trap_mechanism	4637	4558	4682						
check_traps_enabled	4403	4410/P							
check_traps_enabled	4490	4535	4558/P						
check_traps_enabled	4639	4654							
clear	2127	1072/M	2096/M	3904/M	3915/M	3920/M	3929/M	3972/M	3990/M
		3995/M	4004/M	4020/M	4082/M	4104/M	4246/M	4411/M	
cmt$element_state	2333	2292	2293	2323					
condition	213	3704/M	3978/M	4002/M	4200/M	4206/M	4221/M		
condition	3699	3704							
condition	3934	3978							
contents	700	4343	4472	4550					
count	2124	1065/M	1065	1072	1072/M	1072	2093	2094/M	2094

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		2198/M	2198	3900/M	3900	3904	3904/M	3904	3915
		3915/M	3915	3920	3920/M	3920	3928	3928/M	3928
		3968/M	3968	3972	3972/M	3972	3980	3980/M	3980
		3995	3995/M	3995	4004	4004/M	4004	4020	4020/M
		4020	4049/M	4049	4082	4082/M	4082	4095/M	4095
		4104	4104/M	4104	4246/M	4246	4246	4246/M	4246
		4411/M	4411	4411	4411/M	4411			
critical_task	2624	4596							
cst_p	3680	3682/M							
cst_p	3884	3896/M							
cst_p	3891	3896/P	3908	3909/M	3909	3911/M	3911		
cst_p	3934	3964/M							
cst_p	3957	3964/P	3976	3977	3980/M	3980	3982/M	3982	3983/M
		3983	3985/M						
cst_p	4025	4058/M							
cst_p	4045	4058/P	4059	4060/M	4060	4062/M	4062	4063/M	4063
		4065/M							
cst_p	4108	4148/M							
cst_p	4141	4148/P	4150	4151/S	4152	4154/S			
cst_p	4162	4229/M							
cst_p	4184	4229/P	4230						
cst_p	4322	4339/M							
cst_p	4330	4339/P	4340						
cst_p	4361	4377/M							
cst_p	4371	4377/P	4378						
cst_p	4416	4463/M							
cst_p	4458	4463/P	4465	4474	4480/P				
cst_p	4486	4524/M							
cst_p	4514	4524/P	4526	4527	4527	4534	4609/M		
dfc$command_record_bytes	1203	1211							
dfc$division_overwrite_words	1190	1218							
dfc$esm_command_record_size	1211	1219							
dfc$esm_header_record_size	1212	1219							
dfc$esm_maintenance_buf_size	1191	1222							
dfc$esm_memory_base_shift	1197	1219	1220	1220					
dfc$header_record_bytes	1202	1212							
dfc$max_esm_memory_size	1192	1221							
dfc$max_number_of_mainframes	1199	1184							
dfc$min_data_record_bytes	1207	1218							
dfc$min_esm_division_size	1217	1221							
dft$mainframe_set	1184	1134	1135	1311	1312				
dispatching_priority	2639	4150	4152						
dmt$system_file_id	1233	1166							
dpc$console_row_size	1056	1050							
dpc$top_line_message_size	1050	1045	3845						
dpp$display_error	1044	4538	4586	4605					
dpt$top_line_message	3845	3820							
error_message	4460	4466/M	4467/M	4468/P	4478/M	4480/P			
fatal	41	4465							
fault	4331	4342/M	4343	4350/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
fault	4372	4379/M	4383/P						
fault	4459	4471/M	4472	4480/P					
fault	4518	4545/M	4546/M	4547/M	4548/M	4549/M	4550	4616	
fault_contents_p	4516	4550/M	4551/M	4552/M	4553/M	4554/M	4555/M	4556/M	4558/P
		4576/M							
fault_id	4641	4656/M	4673/M						
faults	847	4344/M							
find_free_signal_buffer	4210	4210	4216	4223					
flag_id	940	4284/P							
flag_id	3886	3911	3919	3927					
flag_id	3936	3982	3994	4014					
flag_id	4026	4062	4074						
found	4644	4664/P	4665	4678/P					
free_buffer_loop	4213	4213	4218						
gft$file_descriptor_index	1248	1238							
gft$system_file_identifier	1237	1233	1970	2667	2844	3052			
gft$table_residence	1251	1239							
global_task_id	2628	4151/S	4154/S	4341	4523	4608/P	4620/P	4623/P	4629/P
halt_message	4519	4528/M	4529/M	4530/P	4536/M	4537/M	4540/P	4587/M	4588/M
		4589/P	4597/M	4598/M	4599/P	4603/M	4604/M	4606/P	
hung_task_in_job	1139	4609/M							
i	4043	4050	4051/S	4053	4054/S				
i	4185	4214	4215/S	4227/S	4228/S	4230/S	4231/S		
i	4515	4561/M	4562	4562/S	4562	4562/S	4563/M	4563	4566
		4567/S	4568/S	4569/S	4570/S	4571/S	4572/S		
i#program_error	2104	1072	2091	3904	3915	3920	3928	3972	3990
		3995	4004	4020	4082	4104	4246	4411	
id	2125	1065	1065/M	1072	2090	2192	2196/M	3900	3900/M
		3904	3915	3920	3928	3968	3968/M	3972	3990
		3995	4004	4020	4049	4049/M	4082	4095	4095/M
		4104	4246	4246/M	4246	4411	4411/M	4411	
identifier	692	4342/M	4379/M	4471/M	4545/M	4556	4581		
ijl_thread	3620	4101							
ijle_p	1060	1066	1068/P						
ijle_p	2241	2246	2248/P						
ijle_p	2316	4527	4609/M						
ijle_p	3934	4003	4003/P	4008	4008/P				
ijle_p	3958	3999/P	4003/P	4008/P					
ijle_p	4089	4096							
ijle_p	4162	4246	4246/P						
ijle_p	4186	4191/P	4246/P						
ijle_p	4400	4411	4411/P						
ijle_p	4406	4409/P	4411/P						
in_use	2224	1067	1070/M	1070	2247	2250/M	2250	4003	4003/M
		4003	4008	4008/M	4008	4246	4246/M	4246	4411
		4411/M	4411						
index	863	3893	3912/S	3913/S	3919/S	3926/S	3926/S	3961	3986/S
		3987/S	3994/S	4013/S	4013/S	4015/S	4016/S	4016/S	4053/M
		4066/S	4067/S	4073/S	4073/S	4075/S	4076/S	4076/S	4098
		4098	4099/S	4101/M	4101/S	4151/S	4154/S	4189	4233/S

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		4341	4523					
iot$io_error	2044	1167	2010					
iot$transfer_count	3140	3128						
jdr_p	4520	4627/M	4628					
jmc$detached_job_wait_time_max	2394	2391						
jmc$highest_det_job_wait_time	2404	2394	2405					
jmc$highest_prio_age_interval	2735	2726	2736					
jmc$highest_service_accumulator	1721	1722						
jmc$highest_service_factor_valu	2759	2752						
jmc$highest_working_set_size	2430	2421	2431	2433	2435	2437		
jmc$ies_job_swapped	1479	1488						
jmc$ies_swapin_in_progress	1478	1487						
jmc$iss_idle_tasks_initiated	1494	1521						
jmc$iss_swapin_io_complete	1519	1522						
jmc$iss_swapin_requested	1515	1522						
jmc$iss_swapout_complete	1514	1521						
jmc$iss_swapped_io_cannot_init	1505	1532						
jmc$iss_swapped_no_io	1496	1531						
jmc$keyword_offset_maximum	1738	2422	2727					
jmc$kjl_maximum_entries	1272	1265	1266	1673				
jmc$kol_maximum_entries	1282	1267						
jmc$lock_ajl	1080	1067	1068/P	1070	2247	2248/P	2250	4003	4003/P
		4003	4008	4008/P	4008	4246	4246/P	4246	4411
		4411/P	4411						
jmc$max_active_jobs	1263	2708	2716	2717					
jmc$max_ajl_ord	1264	1257	1263						
jmc$max_dispatching_control	1438	1442							
jmc$max_dispatching_priority	1360	1320	1323	1324					
jmc$maximum_job_classes	1651	1654							
jmc$maximum_job_count	1279	1272							
jmc$maximum_output_count	1289	1282							
jmc$maximum_service_classes	1754	1757							
jmc$min_dispatching_control	1437	1441							
jmc$null_service_class	1747	1748							
jmc$priority_aging_interval_max	2726	2723							
jmc$priority_p1	1374	1321	3577						
jmc$priority_p10	1383	1322							
jmc$priority_p14	1387	1322	3577						
jmc$priority_p8	1381	1321							
jmc$required_offset	1736	2436							
jmc$reserved_ajls	1268	1263							
jmc$service_accumulator_maximum	1713	1710							
jmc$service_factor_value_max	2752	2749							
jmc$system_default_offset	1737	1738	2438						
jmc$system_supplied_name_size	1885	1882							
jmc$unlimited_offset	1734	1723	2395	2406	2432	2737			
jmc$unspecified_offset	1735	2434							
jmc$working_set_size_maximum	2421	2418							
jmp$free_ajl_with_lock	1083	1068	2248	4003	4008	4246	4411		
jmp$unlock_ajl	1059	1074	4246	4411					
jmp$unlock_ajl_with_lock	2240	2253	4003	4008					
jmt$active_job_list	2231	2209							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmt$active_job_list_entry	2223	2231						
jmt$ajl_ordinal	1257	1063	1103	2244	2296			
jmt$delayed_swapin_work	1304	1133	1308					
jmt$detached_job_wait_time	2391	2376						
jmt$dispatching_control	1408	2691						
jmt$dispatching_control_index	1441	1398	1408					
jmt$dispatching_controls	1411	1409						
jmt$dispatching_priority	1320	1115	1399	1400	1401	1413	2288	2639	2641
		3618	3619	3651	4140				
jmt$ijl_block_index	1551	1547							
jmt$ijl_block_number	1550	1546							
jmt$ijl_dispatching_control	1397	1116							
jmt$ijl_entry_status	1474	1102							
jmt$ijl_ordinal	1545	1122	1150	1900	1901	1963	2002	2225	2315
		2366	3610						
jmt$ijl_page_fault_count	1570	1565	1566	1567					
jmt$ijl_page_stats	1564	1560							
jmt$ijl_service_class_stats	1558	1137							
jmt$ijl_statistics	1603	1136							
jmt$ijl_swap_count	1579	1575	1576						
jmt$ijl_swap_counts	1574	1156	1561						
jmt$ijl_swap_status	1492	1105	1106	1107					
jmt$initiated_job_list_entry	1099	1060	1084	1927	2226	2241	2316	2365	3659
		3668	3958	4089	4186	4406			
jmt$input_file_location	1693	1688							
jmt$job_abort_disposition	1702	1686							
jmt$job_class	1654	1161							
jmt$job_control_block	2347	2298							
jmt$job_mode	1657	1118							
jmt$job_priority	1662	1158	1159	2700	2701	2702	2703		
jmt$job_recovery_disposition	1705	1687							
jmt$job_system_id	2410	2362							
jmt$kjl_index	1673	1104	2410						
jmt$maximum_active_jobs	2708	2685							
jmt$priority_aging_interval	2723	2693							
jmt$queue_file_ijl_information	1685	1143							
jmt$scheduling_data	1149	1127							
jmt$scheduling_priority	2699	2692							
jmt$service_accumulator	1710	1151	1152	1153	2683	2684			
jmt$service_class_index	1757	1162	2676	2686					
jmt$service_class_name	2741	2678	2679						
jmt$service_factor_value	2749	2687							
jmt$service_factors	2745	2687							
jmt$swap_data	1165	1129							
jmt$swapout_reasons	1760	1157							
jmt$swapped_job_entry	1775	1174	1928	2385					
jmt$system_supplied_name	1882	1100	2360						
jmt$task_time_slice	1451	1431	1432						
jmt$time_slice_values	1430	1415	2652						
jmt$user_supplied_name	2414	2361							
jmt$working_set_size	2418	2372	2373						
jmv$ajl_p	2209	1067	1070/M	1070	2247	2250/M	2250	4003	4003/M
		4003	4008	4008/M	4008	4246	4246/M	4246	4411

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		4411/M	4411						
job_monitor_taskid	1117	4086	4527						
jsc$isqi_swapped_io_completed	1905	1907							
jsc$isqi_swapped_io_not_init	1904	1907							
jst$changed_asid_entry	1950	1941							
jst$ijl_swap_queue_id	1904	1899							
jst$ijl_swap_queue_link	1898	1111							
jst$io_control_information	1912	1130							
jst$swap_file_descriptor	1926	1131							
jst$swapped_page_descriptor	1935	1933							
jst$swapped_page_descriptors	1932	1929							
lock	1059	1065	1065	1065/M	1065/M	1065			
lock	1059	1072	1072	1072/M	1072	1072/M			
lock	2087	2090	2093	2094/M	2094	2096/M			
lock	2184	2192	2194	2196/M	2198/M	2198			
lock	3884	3900	3900	3900/M	3900/M	3900			
lock	3884	3904	3904	3904/M	3904	3904/M	3915	3915	3915/M
		3915	3915/M	3920	3920	3920/M	3920	3920/M	3929
		3929	3929/M	3929	3929/M				
lock	3934	3968	3968	3968/M	3968/M	3968			
lock	3934	3972	3972	3972/M	3972	3972/M	3990	3990	3990/M
		3990	3990/M	3995	3995	3995/M	3995	3995/M	4004
		4004	4004/M	4004	4004/M	4020	4020	4020/M	4020
		4020/M							
lock	4025	4049	4049	4049/M	4049/M	4049			
lock	4025	4082	4082	4082/M	4082	4082/M			
lock	4087	4095	4095	4095/M	4095/M	4095			
lock	4087	4104	4104	4104/M	4104	4104/M			
lock	4162	4246	4246	4246/M	4246/M	4246			
lock	4162	4246	4246	4246/M	4246	4246/M			
lock	4400	4411	4411	4411/M	4411/M	4411			
lock	4400	4411	4411	4411/M	4411	4411/M			
locked	2123	1065	2194	3900	3968	4049	4095	4246	4411
mcr_fault_p	4332	4343/M	4344/M	4345/M					
mmc$assign_active_null	2884	2885							
mmc$cell_pointer	2983	2988							
mmc$heap_pointer	2984	2992							
mmc$kw_asid	2909	2945							
mmc$kw_clear_space	2907	2932							
mmc$kw_current_segment_length	2906	2926							
mmc$kw_error_exit_procedure	2908	2936							
mmc$kw_gl_key	2908	2930							
mmc$kw_hardware_attributes	2910	2939							
mmc$kw_inheritance	2910	2947							
mmc$kw_max_segment_length	2907	2928							
mmc$kw_preset_value	2909	2934							
mmc$kw_ps_transfer_size	2911	2955							
mmc$kw_ring_numbers	2905	2921							
mmc$kw_segment_access_control	2909	2943							
mmc$kw_segment_number	2906	2924							
mmc$kw_shadow_segment	2911	2949							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$kw_software_attributes	2908	2941							
mmc$kw_wired_segment	2911	2952							
mmc$pq_avail	1792	1838							
mmc$pq_free	1791	1850							
mmc$pq_job_fixed	1832	1839	1851						
mmc$pq_job_working_set	1834	1851	1852						
mmc$pq_shared_first_site	1842	1846							
mmc$pq_shared_num_sites	1843	1846							
mmc$pq_shared_other	1801	1841							
mmc$pq_shared_site_01	1803	1842							
mmc$pq_shared_site_25	1827	1847							
mmc$pq_shared_task_service	1796	1840							
mmc$pq_swapped_io_error	1830	1850							
mmc$pq_wired	1794	1837							
mmc$ring_crossing_offset	32	4675							
mmc$segment_fault_processor_id	772	697							
mmc$sequence_pointer	2983	2990							
mmc$ssk_none	3084	3056							
mmc$ssk_segment_number	3085	3054							
mmp$fetch_stack_segment_info	3672	4663	4677						
mmt$active_segment_table_entry	1960	1938	1976	2009					
mmt$ast_index	1992	1173	1953	2778					
mmt$attribute_keyword	2905	2920							
mmt$global_page_queue_index	1850	2082							
mmt$global_page_queue_list_ent	2072	2082							
mmt$hardware_attribute_set	2974	2940							
mmt$hardware_attributes	2962	2974							
mmt$job_page_queue_index	1851	1777	2083						
mmt$job_page_queue_list	2083	1128							
mmt$link	1983	1961	1999	2000	2069				
mmt$lock_segment_status	3064	2849							
mmt$locked_page	2021	2005							
mmt$max_sdt	2788	2792							
mmt$max_sdtx	2873	2877							
mmt$memory_reserve_request	2050	1121							
mmt$page_age	2028	2008	2032	2032					
mmt$page_frame_index	1921	1913	1915	1916	1917	1985	1985	2052	2053
mmt$page_frame_queue_id	1852	1914	1969	2003					
mmt$page_frame_table_entry	1998	1936	2014						
mmt$page_map_offsets_ord	17	20							
mmt$page_queue_list_entry	2068	2073	2083						
mmt$sdtx_stream_data	2856	2852							
mmt$segment_access_condition	799	698							
mmt$segment_access_rights	3028	2848							
mmt$segment_access_state	3034	2843							
mmt$segment_descriptor	2775	2785	2789						
mmt$segment_descriptor_extended	2841	2870	2874						
mmt$segment_inheritance	2891	2845	2948						
mmt$segment_pointer_kind	2983	2987							
mmt$segment_reservation_state	3074	2846							
mmt$shadow_info	3049	2850							
mmt$shadow_reference_info	3097	2685							
mmt$shadow_segment_kind	3084	3053							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmt$software_attribute_set	2976	2847	2942					
mmt$software_attributes	2970	2976						
mmt$xcb_page_wait_info	3108	2651						
monitor_condition_register	623	4554/M						
monitor_condition_register	3203	4344	4554					
monitor_fault_id	625	4556/M						
monitor_fault_p	4402	4410/P						
monitor_fault_p	4488	4556	4567	4581				
monitor_faults	2656	4562	4562	4567/M	4568/M	4569/M	4570/M	4571/M
		4572/M	4592	4615	4616/M	4617/M		
monitor_flag_id	4088	4100/P						
monitor_flags	2619	3911/M	3911	3983/M	3983	4063/M	4063	
monitor_flags	3615	3919	3926/M	3927	4016/M	4017	4076/M	4077
mtc$scb_max_hardware_status	3853	3812						
mtc$stepped_system	3857	4539	4602					
mtp$cst_p	3680	3896	3964	4058	4148	4229	4339	4377
		4463	4524					
mtp$error_stop	3696	4666						
mtp$set_status_abnormal	3698	3705	3978					
mtp$step_unstep_system	3720	4468	4530	4540	4589	4599	4606	
mtr_flt_message	4489	4529	4537	4588	4598	4604		
mtt$idle_status_block	3790	3777						
mtt$scb_180_status	3766	3757						
mtt$scb_hardware_status	3814	3752	3837					
mtt$scb_hardware_status_count	3812	3815						
mtt$scb_hardware_status_msg	3818	3824						
mtt$scb_hardware_status_msgs	3823	3760						
mtt$scb_hardware_status_options	3804	3814	3823					
mtt$smu_communications_block	3751	3744						
mtt$step_status_block	3785	3778						
mtt$system_idle_update_request	3858	3792	3792					
mtt$system_status_block	3776	3767						
mtt$system_step_update_request	3857	3787	3787					
mtv$cst0	3686	3682	3896	3964	4058	4148	4229	4339
		4377	4463	4524				
mtv$halt_cpu_ring_number	3734	4533						
mtv$scb	3744	4539/M	4602/M					
mtv$sys_core_init_complete	1022	4465	4526					
mtv$system_haltring	3733	4534						
nat$received_message_descriptor	3124	3117	3126					
nat$received_message_list	3116	2633						
new_task_status	3612	3912	3913/M	3986	3987/M	4066	4067/M	
nlc$cc_connect_confirm	3156	3147						
nlc$cc_connect_request	3155	3145						
nlc$cc_expedited_data	3161	3147						
nlc$cc_max_pdu_kind	3163	3166						
nlc$channel_connection_pdu	3179	3131						
nlc$channelnet_pdu	3179	3133						
nlt$cc_pdu_kind	3166	3144						
nlt$cc_seq#_or_connect_time	3143	3132						
nlt$cc_sequence_number	3169	3148						
nlt$device_identifier	3176	3127						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
nlt$pdu_type	3179	3130						
normal	212	3703/M	3895/M	3903	3963/M	3971	3978/M	4001/M
4047/M		4144	4192	4199/M	4205/M	4220/M		
nos_180_status	3757	4539/M	4602/M					
originator	3405	4230/M						
osc$aging_interval_maximum	2442	2445						
osc$base_exception	319	321						
osc$call_instruction	3273	3281						
osc$data_read	3272	3281						
osc$free_flag	652	3910	3981	4061				
osc$free_running_clock_maximum	1009	1006						
osc$invalid_ring	228	268						
osc$max_fault_contents	710	704						
osc$max_idle_count	2580	2588						
osc$max_name_size	809	813	816					
osc$max_number_of_processors	2112	2107	2283					
osc$max_page_frames	1857	1168	1169	1776	1778	1921	1962	2070
osc$max_page_size	2541	2537						
osc$max_page_table_entries	1858	1861						
osc$max_ring	227	268	269					
osc$max_segment_length	251	274	2853	2884				
osc$max_status_condition_code	67	63	79					
osc$max_status_condition_number	3708	3699						
osc$max_string_size	83	86	89	94				
osc$max_tasks	871	868						
osc$maximum_offset	250	251	271	271	272			
osc$maximum_processor_id	3298	3294						
osc$maximum_processor_number	2564	2559						
osc$maximum_processors	2116	2112	2564					
osc$maximum_segment	249	270						
osc$min_ecc	318	319						
osc$min_page_size	2540	2537						
osc$min_ring	226	269						
osc$pr_base_constant	2146	1065	1072	2090	2191	3900	3904	3915
3920		3929	3968	3972	3990	3995	4004	4020
4049		4082	4095	4104	4246	4246	4411	4411
osc$task_time_slice_maximum	1462	1465						
osc$traps_enabled	676	4577	4654	4657				
osk$debug	563	3893	3961	4189	4341	4523		
osk$m	600	3893	3961	4189	4341	4523		
osk$system_class	575	559	560	561	562	563	564	565
osk$unusual	560	4219						
ost$aging_interval	2445	2374	2375					
ost$asid	998	994	1112	1940	1951	1952	1967	2809
2946								
ost$cp_time	1591	1559	1604	2650				
ost$cp_time_value	1589	1154	1592	1593	2369	2370	2663	
ost$cpu_element_id	2556	2314						
ost$cpu_idle_statistics	2583	2317						
ost$cpu_memory_port_mask	2558	2290						
ost$cpu_running_or_stepped	2601	2598	2598					
ost$cpu_state	2596	2299						
ost$cpu_state_reason	2607	2320						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$cpu_state_table	2285	2260	2283	3680	3891	3957	4045	4141	4184
		4330	4371	4458	4514				
ost$cs_lock	3312	2631							
ost$cst_trace_control	3476	2318							
ost$debug_code	3272	3260							
ost$debug_list	3268	3237							
ost$debug_list_entry	3259	3268							
ost$debug_mask	3278	3236							
ost$exchange_package	3186	2618							
ost$execute_privilege	2822	2804	2817						
ost$execution_control_block	2617	2300	2643	3658	3667	3672	3959	4187	4333
		4373	4407	4487	4638				
ost$external_interrupt_request	3484	2306							
ost$family_name	2455	2450							
ost$flags	3243	3193							
ost$frame_descriptor	730	745							
ost$free_running_clock	1006	976	977	1123	1124	1125	1126	1160	1170
		1171	1172	1402	1414	1966	2228	2268	2269
		2368	2377	2649					
ost$global_task_id	862	857	877	939	978	1019	1019	1117	1146
		2295	2364	2628	2629	3357	3405	3588	3643
		3650	3657	3666	3885	3935	4044	4092	4163
		4401							
ost$halfword	1012	2569							
ost$idle_type	2592	2587							
ost$key_lock	257	2810	2931						
ost$key_lock_value	263	260	667	669					
ost$keypoint_class	3254	3206	3256						
ost$keypoint_mask	3256	3209							
ost$logical_processor_id	2559	2291							
ost$minimum_save_area	740	691	715	3198					
ost$monitor_condition	641	648							
ost$monitor_conditions	648	623	720	847	3199	3203			
ost$monitor_fault	687	765	4331	4372	4402	4459	4488	4518	
ost$monitor_fault_contents	704	700							
ost$name	816	833	2414	2453	2455	2677	2741		
ost$p_register	665	43	615	621	741	3187			
ost$page_id	1863	1873							
ost$page_size	2537	2518							
ost$page_table_entry	1868	1877	1937						
ost$page_table_index	1861	1877	2006						
ost$paging_statistics	1627	1605	2658						
ost$parcel	1014	2312	2313						
ost$pre_processed_for_reconfig	3472	2321							
ost$processor_element_id	2568	2556							
ost$processor_element_number	2577	2570							
ost$processor_id	3294	2621	3288						
ost$processor_id_set	3288	2620	3754	3755	3867				
ost$processor_model_number	2467	2461	2571						
ost$processor_serial_number	2545	2462	2572						
ost$pva	279	670	688	848	3231	3249	4334	4334	
ost$rb_system_error	39	4417							
ost$read_privilege	2825	2805	2818						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$real_memory_address	986	2311							
ost$register_number	661	726	734	735	736	3228			
ost$ring	268	280	1038	2807	2808	2842	2922	2923	3248
		4520							
ost$ring1_termination_reason	3306	2654							
ost$segment	270	281	2925	3055	3226	3262	3675	4645	
ost$segment_access_control	2815	2944							
ost$segment_descriptor	2802	2776							
ost$segment_length	274	2927	2929	2951	2953	2956	3676	4646	
ost$segment_offset	271	282	995	2857	3263	3265			
ost$stack_frame_save_area	714	748	828						
ost$state_tables	2283	3686							
ost$status	51	44	616	2938					
ost$status_condition	75	46	213						
ost$status_condition_code	79	54	75						
ost$string	92	55							
ost$string_size	86	93							
ost$system_flag	944	940	3417	3936	4026				
ost$system_virtual_address	993	2011							
ost$task_index	868	863	2039	2040	2325	3607	3620	4043	
ost$task_time_slice	1465	1451							
ost$top_of_stack_pointer	3246	3238							
ost$trap_enable	675	612	3195						
ost$user_condition	651	658							
ost$user_conditions	658	624	718	747	831	3197	3201	3910	3981
		4061							
ost$user_identification	2448	2363							
ost$user_name	2453	2449							
ost$valid_relative_pointer	277	2647	2648						
ost$valid_ring	269	3238	3673						
ost$vector_simulation_control	3865	3756							
ost$virtual_machine_identifier	753	742	3189	3191					
ost$write_privilege	2828	2806	2819						
ost$x_register	662	726	3228						
osv$cpus_logically_on	2107	1065	1072	2089	2190	3900	3904	3915	3920
		3929	3968	3972	3990	3995	4004	4020	4049
		4082	4095	4104	4246	4246	4411	4411	
p	621	4551/M							
p_register	3187	4465	4533	4534	4546	4551	4568	4621	4622
		4628	4663/P	4668	4674	4677/P			
pmc$internal_base_exception	309	313							
pmc$kill_task_flag	944	960							
pmc$max_signal_contents	929	923							
pmc$max_task_id	3366	3363							
pmc$min_ecc	301	297							
pmc$pc_base_exception	297	309							
pme$hung_recipient_task	313	4206							
pmt$binary_mainframe_id	2460	2367							
pmt$condition_identifier	806	800							
pmt$cpu_model_number	2527	2516	2523						
pmt$cpu_serial_number	2530	2517	2522						
pmt$initialization_value	3000	2935							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
pmt$sense_switches	2550	2378						
pmt$signal	885	878	3406	4164				
pmt$signal_contents	923	887						
pmt$signal_id	890	886						
pmt$task_id	3363	2645	3358					
pmt$vector_simulation	3872	3866						
present	763	4562	4562	4572/M	4592	4615	4617/M	
present	3397	4228/M						
ptl_flags	3617	4015	4075	4233				
pva	670	4465	4533	4534	4546	4568	4621	4622	4628
		4663/P	4668	4674	4677/P			
pva	688	4546/M	4568/M					
rb	4109	4143/P	4143/P	4144	4157/P			
rb	4252	4284/P	4284/P	4284/P				
rb	4290	4316/P	4316/P	4316/P				
rb	4417	4465	4467	4475	4476	4477	4478	
readying_task_priority	3619	4151	4154					
readying_task_priority	4140	4152/M	4154/M	4157/P				
requested_status	3786	4539/M	4602/M					
reserved	3398	4215	4227/M					
ring	280	4465	4533	4534	4621	4628	4663/P	4668	4674
		4677/P						
seg	281	4622						
send_monitor_fault	4486	4350	4383	4410	4480	4634		
seqno	864	4054/M	4099/M					
sequence_number	3608	4054	4099					
sft$counter	1637	1606	1607	2379	2381	2382	2384	
sft$file_space_limit_kind	3093	2851						
signal	878	4316/P						
signal	3406	4231/M						
signal	4164	4231						
signals	2657	4215	4227/M	4228/M	4230/M	4231/M		
stack_length	4646	4664/P	4671	4678/P				
stack_segnum	4645	4664/P	4669	4674	4678/P			
status	856	4143/P	4144					
status	876	4316/P						
status	938	4284/P						
status	3611	4051						
status	3700	3703/M	3704/M					
status	3887	3895/M	3902/P	3903				
status	3934	3978/M	3978/M					
status	3937	3963/M	3970/P	3971	3978/P	4001/M	4002/M	
status	4027	4047/M						
status	4093	4100/P						
status	4165	4191/P	4192	4199/M	4200/M	4205/M	4206/M	4220/M	4221/M
		4239/P						
status	4521	4608/P	4620/P	4623/P	4629/P			
status_p	44	4475						
status_p	616	4475/M						
step_status_block	3778	4539/M	4602/M					
syc$ic_fatal_software_error	3490	4468/P	4530/P	4589/P	4599/P			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
syc$ic_software_breakpoint	3503	4540/P	4606/P					
syc$mf_hang_task	3385	4608/P						
syc$mf_invoke_sysdebug	3386	4623/P	4629/P					
syc$ucr_condition	819	830						
syc$user_defined_condition	820	832						
system_error_count	2642	4204	4583/M	4583	4585	4591		
system_flags	2632	3982/M	3982	4062/M	4062			
system_flags	3616	3994	4013/M	4014	4073/M	4074		
system_status	3767	4539/M	4602/M					
system_table_lock_count	2631	4596						
syt$180_idle_code	3484	2307	3721	3768	3770			
syt$monitor_flag	3385	3370	3886	4088				
syt$monitor_flags	3370	2619	3615	3911	3927	3984	4017	4064	4077
syt$monitor_request_code	104	40	855	875	937	974	2266		
syt$monitor_status	211	42	856	876	938	975	2267	3590	3645
		3660	3700	3887	3937	4027	4093	4165	4521
		4647						
task_id	857	4143/P	4157/P					
task_id	877	4316/P						
task_id	939	4284/P						
task_id	3885	3893	3902/P	3908	3912/S	3913/S	3919/S	3926/S	3926/S
		3928/P						
task_id	3935	3961	3970/P	3976	3986/S	3987/S	3994/S	3999/P	4013/S
		4013/S	4015/S	4016/S	4016/S	4019/P			
task_id	4044	4053/M	4054/M	4059	4066/S	4067/S	4073/S	4073/S	4075/S
		4076/S	4076/S	4079/P					
task_id	4163	4189	4191/P	4233/S	4239/P	4241/P			
task_id	4401	4409/P							
task_is_terminating	2634	3977	4000	4198					
taskid	2295	3908	3976	4059	4230	4527			
taskid	4092	4096/M	4098	4098	4099/M	4099/S	4100/P	4101/M	4101/S
text	47	4467	4478						
text_p	45	4476							
text_p	617	4476/M							
tmc$	321	327	330	333	336	339	342	345	348
		351	354	357	360	363	366	369	372
tmc$broken_task_fault_id	772	693	4471	4545	4581				
tmc$btc_invalid_a0	635	619	4673						
tmc$btc_invalid_p	635	619							
tmc$btc_mcr_traps_disabled	636	620							
tmc$btc_mf_traps_disabled	635	618	4656						
tmc$btc_mntr_fault_buffer_full	634	618	4576						
tmc$btc_system_error	637	614	4473						
tmc$btc_ucr_traps_disabled	636	620							
tmc$dummy_fault	773	699							
tmc$flag_available_31	957	961							
tmc$maximum_monitor_faults	777	768	4515	4562	4562	4566			
tmc$maximum_signals	3414	3411	4185	4214					
tmc$maximum_system_task_id	3422	3425							
tmc$mcr_fault	772	695	4342						
tmc$mf_cause_job_free_flag_trap	3385	3984	4017	4064	4077	4239/P	4620/P		
tmc$opt_return	3626	3902/P	3970/P	4143/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$rc_ready_conditional	3573	3928/P	4241/P						
tmc$rc_ready_conditional_wi	3573	4019/P	4079/P	4157/P					
tmc$signal_available_63	921	932							
tmc$stid_null_task	3428	3425							
tmc$ts_first_ready_uncond	3547	3912	3986	4066					
tmc$ts_io_wait_not_queued	3565	3547							
tmc$ts_null	3552	3913	3987	4051	4067				
tmc$ts_page_wait	3567	3548							
tmc$ts_ready	3552	3543							
tmc$ts_timed_wait_not_queued	3559	3546							
tmc$ts_timeout_reqexp_longvlong	3557	3545							
tmc$ts_timeout_reqexp_shortshrt	3556	3544							
tmc$unknown_system_req_fault	773	4379							
tmc$wi_wait_selected_r3	3638	4015	4075	4233					
tme$invalid_global_taskid	327	3978/P	4002	4200					
tme$mtr_signal_buffers_full	345	4221							
tmk$monitor_base	535	388	391	394	397	400	403	406	409
		412	415	418	421	424	427	431	434
		437	440	444	448	451	454	457	460
		463	466	469	472	475	478	481	484
		487	490	493					
tmk$process_task_mcr_fault	484	4341							
tmk$send_monitor_fault	472	4523							
tmk$send_signal	469	4189							
tmk$set_monitor_flag	490	3893							
tmk$set_system_flag	466	3961							
tmk$signal_buffers_full	481	4219							
tmp$check_taskid	3588	4143							
tmp$check_taskid_with_lock_set	3642	3902	3970						
tmp$clear_lock	2087	1072	2100	3904	3915	3920	3929	3972	3990
		3995	4004	4020	4082	4104	4246	4411	
tmp$find_xcb	3657	4191							
tmp$flag_all_tasks	4025	4084							
tmp$get_xcb_p	3666	3999	4409						
tmp$monitor_flag_job_tasks	4087	4105							
tmp$mtr_process_system_error	4416	4483							
tmp$mtr_ready_task	4108	4159							
tmp$mtr_send_signal	4289	4318							
tmp$mtr_set_system_flag	4251	4286							
tmp$process_task_mcr_fault	4322	4357							
tmp$process_unknown_req_fault	4361	4385							
tmp$send_monitor_fault	4400	4413							
tmp$send_signal	4162	4248	4316						
tmp$set_lock	2184	1065	2202	3900	3968	4049	4095	4246	4411
tmp$set_monitor_flag	3884	3931	4100	4239	4608	4620	4623	4629	
tmp$set_system_flag	3934	4022	4284						
tmp$set_task_ready	3650	3928	4019	4079	4157	4241			
tmt$broken_task_condition	634	613	4641						
tmt$broken_task_monitor_fault	611	694	4461	4516					
tmt$dispatch_control	3514	2302							
tmt$dual_state_priority_entry	3581	2289	3578						
tmt$idle_status	3570	3613							
tmt$mcr_faults	846	696	4332						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmt$monitor_fault_buffer	762	2656							
tmt$monitor_fault_buffers	768	763	764	765					
tmt$monitor_fault_identifiers	771	625	692						
tmt$option	3626	3589	3644						
tmt$primary_task_list	3624	3737							
tmt$primary_task_list_entry	3606	3624							
tmt$ptl_flags	3630	3617							
tmt$ptl_lock	2120	2087	2184	2235					
tmt$rb_delay	2265	2259							
tmt$rb_ready_task	854	4109							
tmt$rb_send_signal	874	4290							
tmt$rb_set_system_flag	936	4252							
tmt$rb_wait	973	981							
tmt$ready_condition	3573	3652							
tmt$signal	3404	3399							
tmt$signal_buffer	3396	2657							
tmt$signal_buffers	3411	3397	3398	3399					
tmt$system_flags	3417	2632	3616	3982	4014	4062	4074		
tmt$system_task_id	3425	2623							
tmt$task_queue_link	2038	2007	3614						
tmt$task_status	3552	3517	3611	3612					
tmt$wait_inhibited	3637	3632							
tmt$xcb_offset_size	3628	3609							
tmv$halt_on_hung_task	1035	4601							
tmv$job_debug_ring_p	1038	4626	4627						
tmv$ptl_lock	2235	1065/P	1072/P	3900/P	3904/P	3915/P	3920/P	3929/P	3968/P
		3972/P	3990/P	3995/P	4004/P	4020/P	4049/P	4082/P	4095/P
		4104/P	4246/P	4246/P	4411/P	4411/P			
tmv$ptl_p	3737	3912	3913/M	3919	3926/M	3926	3986	3987/M	3994
		4013/M	4013	4015	4016/M	4016	4050	4051	4054
		4066	4067/M	4073/M	4073	4075	4076/M	4076	4099
		4101	4151	4154	4233				
tmv$system_debug_ring	1036	4621							
tmv$system_debug_segment	1037	4622	4622						
tmv$system_error_hang_count	1039	4204	4585	4591					
trap_enable	612	4474/M	4553/M						
trap_enable	3195	4474	4553	4577/M	4654	4657/M			
untranslatable_pointer	848	4345/M							
untranslatable_pointer	3231	4345	4355/M						
user_condition_register	624	4555/M							
user_condition_register	3201	3909/M	3910	3980/M	3981	4060/M	4061	4555	
wait_inhibited	2630	3985/M	4065/M	4238/M					
wait_inhibited	3632	4015	4075	4233					
xcb_p	2300	3909/M	3909	3911/M	3911	3977	3980/M	3980	3982/M
		3982	3983/M	3983	3985/M	4060/M	4060	4062/M	4062
		4063/M	4063	4065/M	4150	4151/S	4152	4154/S	4340
		4378	4465	4474	4480/P				
xcb_p	3959	3999/P	4000	4000	4007				
xcb_p	4187	4191/P	4198	4204	4215	4227/M	4228/M	4230/M	4231/M
		4238/M							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	4333	4340/M	4341	4344	4345	4350/P	4355/M		
xcb_p	4373	4378/M	4383/P						
xcb_p	4407	4409/P	4410/P						
xcb_p	4487	4523	4533	4534	4546	4547	4548	4549	4551
		4552	4553	4554	4555	4558/P	4562	4562	4567/M
		4568/M	4568	4569/M	4569	4570/M	4570	4571/M	4571
		4572/M	4577/M	4583/M	4583	4585	4591	4592	4596
		4596	4608/P	4615	4616/M	4617/M	4620/P	4621	4622
		4623/P	4627	4628	4629/P				
xcb_p	4638	4654	4657/M	4663/P	4663/P	4668	4668	4669	4670
		4671	4674/M	4674	4676/M	4677/P	4677/P		
xp	2618	3909/M	3909	3980/M	3980	4060/M	4060	4344	4345
		4355/M	4465	4474	4533	4534	4546	4547	4548
		4549	4551	4552	4553	4554	4555	4568	4569
		4570	4571	4577/M	4621	4622	4628	4654	4657/M
		4663/P	4668	4668	4669	4670	4671	4674/M	4674
		4676/M	4677/P						
zero_pva	4334	4355							

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

CONTROL DATA

4201 North Lexington Avenue, Saint Paul, Minnesota 55126-6198