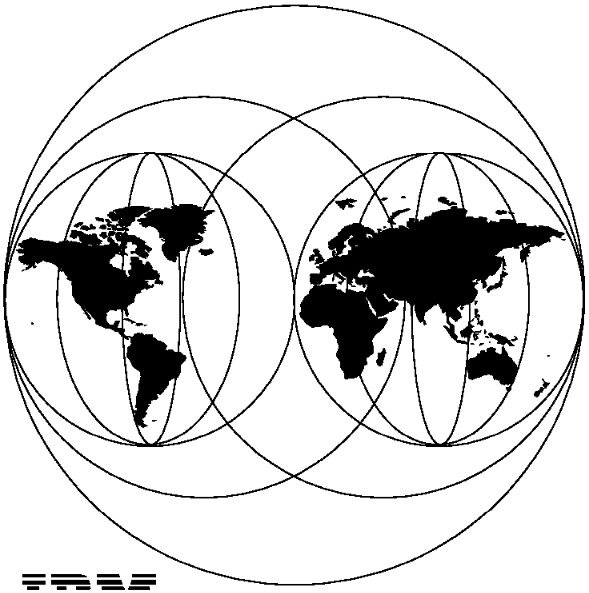
International Technical Support Organization SG24-4643-00

OS/2 Debugging Handbook - Volume IV
System Diagnostic Reference

February 1996



IBM

International Technical Support Organization
Boca Raton Center

International Technical Support Organization

SG24-4643-00

OS/2 Debugging Handbook - Volume IV System Diagnostic Reference

February 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xvii.

First Edition (February 1996)

This edition applies to IBM OS/2 Warp Version 3.0.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. JLPC Building 014-1 Internal Zip 5220
1000 NW 51st Street
Boca Raton, Florida 33431-1328

The OS/2 Debug Handbook Library

The following information describes the four volumes that comprise the OS/2 Debug Handbook library. The graphic of the opened book denotes the volume that you are currently reading.

Volume I, *Basic Skills and Diagnostic Techniques, SG24-4640.*

This volume introduces the concepts of debugging with practical examples. Also contained in this book is a CDROM version of the entire library, which is viewable via the OS/2 INF View utility.

Volume II, *Using the Debug Kernel and Dump Formatter, SG24-4641.*

This volume provides necessary information to set up and use the Kernel Debug and Dump Formatter tools. Also this guide serves as a command reference for these products.

Volume III, *System Trace Reference, SG24-4642.*

This volume includes all system tracepoints contained within OS/2.

Volume IV, *System Diagnostic Reference, SG24-4643.*

This volume provides details of internal structures used by OS/2.

Abstract

This publication is volume four, which is one of four volumes that together provide information and reference materials intended to help perform OS/2 debugging.

This volume provides system reference and control block information that is used within OS/2. It is intended that this volume be used *only* in conjunction with the other volumes in the library.

This document is intended for use by service personnel, system programmers and software developers.

(286 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help service personnel, system programmers and software developers to understand the concepts and application of debugging techniques. The information in this publication is intended as a supplement to already published specifications of any programming interfaces that are provided by IBM Warp OS/2 Version 3. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Warp OS/2 Version 3 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM
Presentation Manager
OS/2
Workplace Shell

The following terms are trademarks of other companies:

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Windows is a trademark of Microsoft Corporation.

MicroFocus Cobol	MicroFocus Corporation

Other trademarks are trademarks of their respective companies.

Preface

This volume of the OS/2 Debugging Handbook Library is a reference to the Reliability Availability and Serviceability interfaces, OS/2 system control blocks and other reference tables. This book should *only* be used in conjunction with the other volumes in this library.

This document is intended for use by service personnel, system programmers and software developers.

How This Document is Organized

The document is organized as follows:

- Chapter 1, "CONFIG.SYS RAS Commands"

  Details of the commands are in this chapter if they are not mentioned in the OS/2 Command Reference manual.

- Chapter 2, "OS/2 RAS Application Programming Interfaces"

  Information pertaining to the application programming interfaces are described in this chapter.

- Chapter 3, "OS/2 System Control Block Reference"

  This large section documents major control blocks that are defined by the base OS/2 system.

- Chapter 4, "Reference Tables"

  This section documents various system tables, system error codes and other miscellaneous tables.

Related Publications

Throughout this book we assume the availability and familiarity with three co-requisite publications:

- *The INTEL486 Microprocessor Programmer's Reference Manual*, ISBN 1-55512-159-4
- *The Intel Pentium Family User's Manual, Volume 3: Architecture and Programming Manual*, ISBN 1-55512-227-2
- *The Design of OS/2 by H.M. Deitel and M.S. Kogan*, ISBN 0-201-54889-5

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *The OS/2 Technical Library Control Program Programming Reference Version 2.00*, S10G-6263-00
- *OS/2 2.0 Proc Lang 2/REXX Ref*, S10G-6268-00
- *OS/2 2.0 Proc Lang 2/REXX User Guide*, S10G-6269-00
- *OS/2 WARP Control Program Programming Guide*, G25H-7101-00
- *OS/2 WARP Control Program Programming Ref*, G25H-7102-00
- *OS/2 WARP PM Basic Programming Guide*, G25H-7103-00

- *OS/2 WARP PM Advanced Programming Guide*, G25H-7104-00
- *OS/2 WARP GPI Programming Guide*, G25H-7106-00
- *OS/2 WARP GPI Programming Ref*, G25H-7107-00
- *OS/2 WARP Workplace Shell Programming Guide*, G25H-7108-00
- *OS/2 WARP Workplace Shell Programming Ref*, G25H-7109-00
- *OS/2 WARP IPF Programming Guide*, G25H-7110-00
- *OS/2 WARP Tools Reference*, G25H-7111-00
- *OS/2 WARP Multimedia App Programming Guide*, G25H-7112-00
- *OS/2 WARP Multimedia Subsystem Programming*, G25H-7113-00
- *OS/2 WARP Multimedia Programming Ref*, G25H-7114-00
- *OS/2 WARP PM Programming Ref Vol I*, G25H-7190-00
- *OS/2 WARP PM Programming Ref Vol II*, G25H-7191-00
- *Technical Reference - Personal Computer AT*, Part Number 1502494
- *PS/2 and PC BIOS Interface Technical Reference*, Part Number 68X2341

International Technical Support Organization Publications

- *OS/2 Warp Connect*, GG24-4505
- *OS/2 Warp Generation, Vol.1*, SG24-4552
- *OS/2 Warp Version 3 and BonusPak*, GG24-4426
- *Multimedia in Warp*, GG24-2516
- *The Technical Compendium Volume 1 - Control Program*, GG24-3730
- *The Technical Compendium Volume 2 - DOS and Windows Environment*, GG24-3731
- *The Technical Compendium Volume 3 - Presentation Manager and Workplace Shell*, GG24-3732
- *The Technical Compendium Volume 4 - Application Development*, GG24-3774

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

> *International Technical Support Organization Bibliography of Redbooks,* GG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOCAT TXT. This package is updated monthly.

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-445-9269. Most major credit cards are accepted. Outside the USA, customers should contact their local IBM office. Guidance may be obtained by sending a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following URL:

`http://www.redbooks.ibm.com/redbooks`

IBM employees may access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks home page:

`http://w3.itsc.pok.ibm.com/redbooks/redbooks.html`

Acknowledgments

The authors of this book are:

Pete Guy
IBM SDO, Austin

Richard Moore
IBM PSP EMEA

Redbook project developed by:

Tim Sennitt
ITSO Boca Raton, Center

This book could not have reached publication without the encouragement, help and support from a number of colleagues and friends. In particular we would like to thank the following:

Tim Sennitt for his help in preparing the printed material and doing much of the donkey-work to bring this to publication.

Joanne Rearnkham, Barry Bryan and David Jaramillo for their support in enabling access to the materials necessary to produce this book.

Chris Perritt and Glen Brew for making available the original Design Workbook and Functional Specifications for OS/2 2.0.

Charlie Schmitt for his original work on converting the kernel debugger code into a dump formatter.

Jeff Mielke and David Jaramillo for their work on PMDF, the structure compiler and continued work on the dump formatter.

Allen Gilbert for making available documentation on System Trace, which has been reproduced in an edited form in this book. Also, for making available an early version of the dump formatter without which it would not have been possible to develop the original Dump Formatter class.

Doug Azzarito for supplying the material on Kernel Debugger Remote Debug Setup.

James Taylor for providing the basis of the lab exercises relating to PM hangs.

Marie Jazynka, one of the first OS/2 debuggers, for patient encouragement of a great many OS/2 debugging people.

Our management teams, without whose foresight and support none of this work would ever have started. These include:

- Hermann Lamberti General Manager for PSM EMEA; Gordon Bell - director PSM EMEA Technical Marketing; Chris Brown - manager PSM OEM and Enterprise Technical Marketing and Brian Rose - manager PSM Project Office; Roy Aho - Director of the Solution Developer Technical Support Center, for encouraging the beginnings of this several years ago; Terry Gray, manager of Platform Competency and Operation, within Solution Developer Technical Support, Austin.

Finally to Sarah-Jane and Shelly, for supporting many very extended working days and weeks.

Chapter 1. CONFIG.SYS RAS Commands

OS/2 provides a number of CONFIG.SYS commands and settings specifically for RAS purposes. Some of these are described in the *OS/2 Command Reference* and will not be discussed in detail here. A number of commands were introduced or enhanced in APAR PJ12258, which is applicable to OS/2 2.11. All the commands described here are available with OS/2 Warp 3.0.

The following CONFIG.SYS commands comprise the RAS set. Those not completely documented in the OS/2 Command Reference are now discussed in detail.

- AUTOFAIL (see *OS/2 Command Reference, autofail*)
- 1.2, “DUMPPROCESS”
- 1.3, “REIPL” on page 2
- 1.4, “SHAPIEXCEPTIONHANDLER” on page 2
- 1.5, “SHELLEXCEPTIONHANDLER” on page 3
- 1.6, “SUPPRESSPOPUPS” on page 4
- TRACE (see the *OS/2 Command Reference*). Also see the *System Trace User Guide in Volume 3 of The OS/2 Debugging Library.*
- TRACEBUF (see the *OS/2 Command Reference*).
- TRACEFMT (see the *OS/2 Command Reference*).
- See also 1.10, “TRAPDUMP” on page 5.

1.1 AUTOFAIL

AUTOFAIL modifies the processing of media errors. See the *OS/2 Command Reference, AUTOFAIL command*.

1.2 DUMPPROCESS

This command allows the user to activate the process dump facility. When active, any ring 3 (application) process that traps will result in a memory dump being written to a unique dump file. The dump file takes a name of the form *PDUMP.nnn* where nnn is an index that is incremented each time a new process dump is created.

The contents of the dump comprise unformatted system and user storage that relates to the trapping process. Included in this are:

- PTDA
- TCB and TSD
- Registers
- Arena records
- MTE and SMTEs
- LDT
- Ring 0 stack

- Ring 3 stack

Syntax

```
DUMPPROCESS=x
```

Parameters

x This specifies the drive letter (excluding the colon) to which process dump data sets will be written. These take the name *PDUMP.nnn* and reside in the root directory of the drive specified. The name and directory cannot be overridden by the user.

Note: See the Process Dump Formatter section of the *Dump Formatter User Guide* for information on formatting process dumps.

1.3 REIPL

The REIPL command allows the user to automate the re-booting (re-IPLing) of the system following an IPE.

Syntax

```
REIPL=ON | OFF
```

Parameters

ON
This specifies the system to be automatically re-booted following an IPE.

OFF
This specified that the system is not to be automatically re-booted following an IPE. The system will remain hung until manually restarted.

Notes: REIPL only applied to pre-WARP systems if APAR PJ12258 has been applied.

REIPL has no effect when TRAPDUMP=ON|R0 is specified. Whether the system is re-booted following a stand-alone dump is governed by the OS2DUMP module. If the the dump is to hard-disk then automatic re-boot occurs, otherwise not.

1.4 SHAPIEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the PMSHAPI.DLL module.

Syntax

```
SET SHAPIEXCEPTIONHANDLER=ON|OFF
```

Parameters

ON

This is the default setting. The shell API DLL exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

OFF

The shell API DLL exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

Notes: Exception handler registration only occurs during PMSHAPI.DLL initialization. Therefore, a change to the specification of SHAPIEXCEPTIONHANDLER will require the system to be re-booted.

The shell API DLL exception handler will attempt to clean up an application′s PM resources.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analyzed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. SHAPIEXCEPTIONHANDLER may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

SHAPIEXCEPTIONHANDLER may be used with TRAPDUMP to force a system dump at the point of failure.

Hangs in the shell during initialization may be the result of a recursive trap. SHAPIEXCEPTIONHANDLER may be used to intercept this condition.

Since it is difficult to determine whether a potential shell problem involves PMSHELL.EXE or PMSHAPI.DLL, it is recommended to use SHAPIEXCEPTIONHANDLER with SHELLEXCEPTIONHANDLER.

1.5 SHELLEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the PMSHELL.EXE module.

Syntax

```
SET SHELLEXCEPTIONHANDLER=ON|OFF
```

Parameters

ON

This is the default setting. The shell′s exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

OFF

The shell′s exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

Notes: Exception handler registration only occurs during PMSHELL.EXE initialization. Therefore, a change to the specification of SHELLEXCEPTIONHANDLER will require the system to be re-booted.

The shell′s exception handler will attempt to clean up an application′s PM resources. In addition if the application is the desktop (or whatever is specified in RUNWORKPLACE), then it is restarted.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analyzed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. SHELLEXCEPTIONHANDLER may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

SHELLEXCEPTIONHANDLER may be used with TRAPDUMP to force a system dump at the point of failure.

Hangs in the shell during initialization may be the result of a recursive trap. SHELLEXCEPTIONHANDLER may be used to intercept this condition.

Since it if difficult to determine whether a potential shell problem involves PMSHELL.EXE or PMSHAPI.DLL, it is recommended to use SHELLEXCEPTIONHANDLER with SHAPIEXCEPTIONHANDLER.

1.6 SUPPRESSPOPUPS

This command allows the user to suppress the display of trap information pop-up messages and instead, direct trap information to a log data set.

Syntax

```
SUPPRESSPOPUPS=x
```

Parameters

x This specifies the drive letter (excluding the colon) to which the pop-up log data set will be written. The log takes the name *POPUPLOG.OS2* and resides in the root directory of the drive specified. The name and directory cannot be overridden by the user.

1.7 TRACE

TRACE specifies whether tracing of static trace events is to be active from system initialization or not. See the *OS/2 Command Reference*, TRACE command for details. Also see the *System Trace User Guide* in Volume 3 of The OS/2 Debugging Library.

1.8 TRACEBUF

TRACEBUF specifies the size of the system trace buffer. See the *OS/2 Command Reference*, TRACEBUF command for details.

1.9 TRACEFMT

The TRACEFMT utility is used to extract and format the system trace from the either a saved trace buffer or the currently active trace buffer. See the *OS/2 Command Reference*, TRACEFMT command for details.

1.10 TRAPDUMP

Potential Data Loss

Misuse of this facility may cause loss of vital data. Please read carefully the complete description before use.

The TRAPDUMP command controls the stand-alone (system) dump facility of OS/2. It will enable initiation of a stand-alone dump at the instant a ring 3 trap occurs for which no exception handler has intervened.

Ring 0 traps may be also intercepted only on 2.11 systems to which APAR PJ12258 has been applied, or on OS/2 Warp.

Pre-Warp considerations:
The dump process is performed by the hidden module OS2DUMP, which resides in the root directory of the boot drive. OS2DUMP as supplied with GA versions of OS/2 2.x dumps only to diskette. It may be replaced with a version supplied with OS/2 Problem Determination Package (OS2PDP) which will dump to a hard disk FAT partition that has the volume label SADUMP or to diskette, depending upon TRAPDUMP command specification.

The GA 2.x version of OS2DUMP requires the first dump diskette be freshly prepared using the `CREATEDD` command and subsequent diskettes to be formatted. See the on-line *OS/2 Command Reference* for details of `CREATEDD` command.

The OS/2 Problem Determination Package (OS2PDP) version of OS2DUMP only requires formatted diskettes, the use of `CREATEDD` being redundant.

When dumping to hard disk the dump partition must to be made known to TRAPDUMP. This is done by specifying an optional second parameter.

OS/2 Warp considerations:
Under OS/2 Warp the `CREATEDD` command is unnecessary and is not distributed with the system. Ordinarily formatted diskettes may be used. Furthermore the enhanced version of OS2DUMP which allows dumping to a hard-disk FAT partition is standard. The partition volume label must be SADUMP.

Syntax

```
TRAPDUMP=[ON|OFF|RO][,]X:
```

Parameters

ON

Specifies that the stand-alone dump process will be automatically initiated whenever an unrecoverable ring 3 trap occurs. For 2.11 systems with APAR PJ12258 or OS/2 Warp, any system IPE (including ring 0 traps) will also initiate a dump when **ON** is specified.

OFF

Specifies that the stand-alone dump process will not initiate automatically when an unrecoverable trap occurs. This is the default option. It does not prohibit the use of the Ctrl-Alt-Numlock-Numlock key sequence or the use of DosForceSystemDump to force a stand-alone dump to be initiated.

R0 Specifies that only ring zero traps and IPEs will automatically initiate the stand-alone dump process. This option applies only to 2.11 systems with APAR PJ12258 or OS/2 Warp.

Note: When an IPE occurs the dump is taken immediately on displaying the IPE trap screen. For the purposes of dump analysis the formatted registers from the IPE screen should be located from the video buffer, which may be viewed using the analyze option from the PMDF.

X: specifies the hard-disk FAT partition to which OS2DUMP will write a stand-alone dump. The partition letter must have the colon suffix.

Note: The partition may be specified with either ON or OFF. When specified with OFF it will allow a stand-alone dump initiated by Ctrl-Alt-Numlock-Numlock to be written to the dump partition.

Mountable media other than diskette drives are not detectable by OS2DUMP. The letter specifying the dump partition must be calculated as if any such media were *not* present.

Only hard disk logical drives and primary partitions may be specified.

When dumping to a hard disk partition is selected the system is automatically re-booted on completion of the dump.

Attention

The stand-alone dump process will erase all data on the dump media (disk partition or diskettes) before writing the dump.

Do not specify a disk partition or use diskettes that contain vital data.

Chapter 2. OS/2 RAS Application Programming Interfaces

This chapter describes the subset of OS/2 RAS APIs for use by application programmers, which are not described in the *OS/2 Technical Library, Control Programming Reference.*

Caution

Some RAS programming interfaces may be specific to a particular release of OS/2 or have a release specific function.

The APIs discussed in this section are:

2.1 DosSysTrace (Static Trace Event Recording)

Static trace recording is available as both an API and a DevHlp routine.

2.1.1 DosSysTrace (Add a Trace Record to the System Trace Buffer)

DosSysTrace allows a subsystem or system extension to add information to the system trace buffer.

Note: DosSysTrace is a 16-bit API.

Coding Examples

```
EXTRN  DosSysTrace:FAR

PUSH     WORD     MajorCode     ; major trace event code (240-255)
PUSH     WORD     Length        ; length of the variable length
                                ; area to be recorded (0-512)
PUSH     WORD     MinorCode     ; minor trace event code (0-255)
PUSH@    OTHER    Data          ; pointer to the area to be traced
                                ; (address parameter)
CALL     DOSSYSTRACE
```

16-bit MASM Example

```
APIRET16 APIENTRY16 DosSysTrace(USHORT MajorCode, USHORT Length,
                                USHORT MinorCode, PCHAR pData);
```

32-bit code Example using CSet/2

Parameters

MajorCode
The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

Length
The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.

MinorCode
The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

pData
The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

Results

DosSysTrace returns the following values:

0 NO_ERROR

150
ERROR_SYSTEM_TRACE (trace is disabled for that event)

```
IF AX = 0
      Data traced
ELSE
   AX = Error_System_Trace
      Data not traced
```

Note: An example of when data would not be traced is if the major event code is not currently selected for tracing.

Remarks

All trace records consist of a header and optional data. The header record is built by DosSysTrace and contains:

- Major event code
- Minor event code
- Process ID of caller
- Timestamp when the time is different from the previous trace record
- Flag field
- Data field (optional)

The optional data field contains the variable-length data as passed by the caller.

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The trace facility checks this array each time it is called to ensure that the major event specified is currently enabled for tracing. The array is located in the Global Information Segment.

A prototype definition for DosSysTrace may be found under 2.9, “RAS API Prototypes” on page 37.

2.1.2 DevHlp_SysRAS (Add a Trace Record to the System Trace Buffer)

The DevHlp_SysTrace function provides a service for device drivers to add information to the system trace buffer.

Note: DevHlp_SysTrace is a 16-bit API.

Coding Example

```
MOV   AX,MajorCode          ; major trace event code (240-255)
MOV   BX,Length             ; length of data area (0-512 bytes)
MOV   CX,MinorCode          ; minor trace event code (0-255)
LDS   SI,pData              ; pointer to trace data
MOV   DL,28H                ; DevHlp_SysRAS function code
CALL  [Device_Help]         ; invoke device helper
```

16-bit MASM Example

Parameters

MajorCode

The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

Length

The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.

MinorCode

The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

pData

The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

Results

```
If CF = 0
   Trace record placed in trace buffer
Else
   Data not traced
```

The possible errors are as follows:

- Tracing suspended
- Minor code not being traced
- PiD not being traced
- Trace overrun

Remarks

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The device driver must check this array before calling DevHlp_SysTrace to ensure that the major event specified is currently enabled for tracing. This array is located in the Global Information Segment.

All registers are preserved. Interrupts are disabled while the trace data is saved and then re-enabled if they were initially enabled.

2.2 DosGetSTDA (Get the System Trace Data Area)

The DosGetSTDA API is a 16-bit API that returns a copy of the system trace buffer (STDA).

Syntax

The following 16-bit C language function prototype can be used to call the DosGetSTDA API:

```
// 16 bit compiler
extern unsigned far pascal DosGetSTDA( SEL, SHORT, SHORT );

// 32 bit compiler
APIRET16 APIENTRY16 DosGetSTDA( SEL, SHORT, SHORT );

      Where:  SEL    is the selector to the private buffer
              SHORT  is the offset to the private buffer
              SHORT  is the size of the buffer
                     (maximum value = 64KB)
                       records

      Returns:  0 - indicates correct operation, buffer is now filled
                    with copy of the system trace buffer
                ERROR_SYSTEM_TRACE - System trace is not enabled
```

Linker Considerations

In order to successfully resolve DosGetSTDA function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSGETSTDA=DOSCALL1.119
```

Remarks

DosGetSTDA returns a buffer that contains a copy of the system trace buffer. The buffer is circular with a header record that contains pointers to the first and last data bytes and a pointer to the next byte that was available for writing (the buffer is a snapshot of the system trace buffer at the time that the API was called). A set of trace records follows the header. Each trace record contains a trace event trailer and optionally a timestamp and/or a data field. A timestamp record is optional and will only exist if bit 2 of the flag field in the Trace Event Trailer is set to OFF.

The trace event data contains the information describing each individual trace event. The events traced may be from OS/2 system supplied or other user supplied trace points. In either case the data is dependent on each individual trace point. Descriptions of the data and formatting instructions for the OS/2 system supplied trace points can be found in the *OS/2 Debugging Library, Volume 3 - System Trace Points Reference.*

2.2.1 Trace Buffer Structures

Note: From the OS/2 2.11 FixPak 91 and OS/2 3.0 FixPak 8 the format of the STDA has changed to allow more meaningful timestamp information. See 2.2.1.1, “New STDA Format” on page 14 at the end of this section for details.

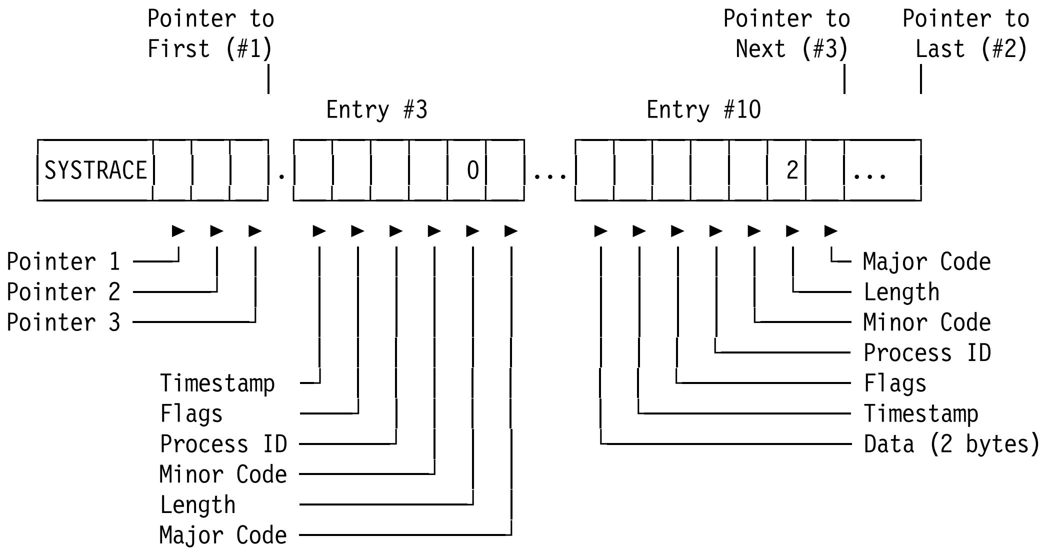


Figure 1. Circular Trace Buffer (STDA)

Table 1. Field Descriptions: Trace Control Record		
Name	**#Bytes**	**Description**
ID of Data Area	8	Contains ASCII ′SYSTRACE′
Pointer 1	2	Offset of first byte of the trace buffer
Pointer 2	2	Offset of last byte of the trace buffer
Pointer 3	2	Offset of next available byte in the trace buffer

Table 2. Field Descriptions: Trace Event Trailer Record (with Timestamp)		
Name	**#Bytes**	**Description**
Timestamp	(Conditional on bit 1 in the flags byte)	Timestamp in seconds and hundredths of seconds (Conditional on bit 2 in the Flags byte)
Flags	1	Trace record flag Bit 0: 0 indicates an internal kernel generated trace record. Bit 1: 0 indicates that a timestamp is present. Bit 2: 1 signifies that the trace record was generated in protect mode. Bit 3: 0 signifies a static trace record, 1 a dynamic trace record. Bit 4: 1 indicates an incomplete dynamic trace record. Bit 5 - 7: reserved.
PID	2	ID of the process calling the API being traced
Minor Code	2	Minor Event Code
Length	2	Length of data for the traced API
Major Code	1	Major Event Code

Remarks

The buffer returned by DosGetSTDA is a simple circular buffer that is a snapshot of the OS/2 System Trace buffer at the time that the API was called. The actual System Trace buffer is emptied by the call. The buffer contains a header record that has pointers to the *First*, *Last* and *Next* bytes in the buffer. The offsets of the *First* and *Last* bytes are constant and the offset to *next* is used to indicate the last (most recent) trace record in the buffer. This pointer is logically moved backwards as the buffer is traversed. Since it is possible for a trace record to wrap back to the end of the buffer, it is necessary to look at each part of the data individually (trailer, timestamp and data) to determine whether the length of the data is greater than the distance between *Next* and *First*. If the length is greater, then the data is continued at the offset to *Last*.

For example (see figure below), the buffer has been traversed until the pointer to *Next* is at byte 26. The event trailer record is 8 bytes and the distance from *Next* to *First* is 12, so the trailer is in contiguous memory. The pointer to *Next* is then set to byte 18. There is a timestamp which is two bytes. Our distance to *First* is now 4 so the timestamp is contiguous and the pointer to *Next* is reset to 16. This record has 4 bytes of data attached to it. The distance to *First* at this point is 2, so the data is wrapped: 2 bytes are adjacent to the *Next*, and the other 2 bytes begin at the pointer to *Last*.

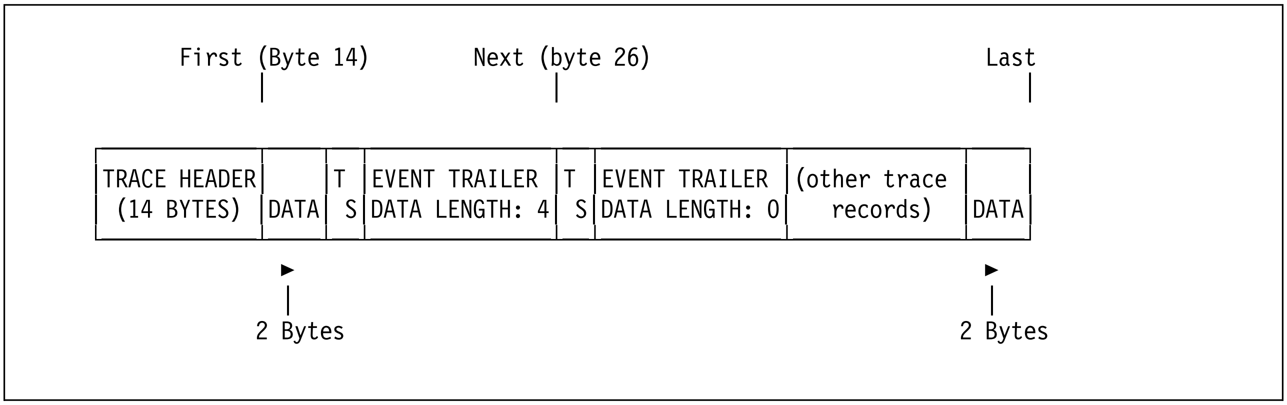


Figure 2. Buffer Returned by DosGetSTDA

The end of data in the trace buffer is indicated by a trace event trailer that contains a major code field of zero and a length field of zero.

The display format of the OS/2 system supplied tracepoint data is described in the *OS/2 Debugging Library, Volume 3 - System Trace Points Reference.* Note that for data using the %S (ASCIIZ string) format type, the first byte of the data is reserved, bytes two and three contain the actual length of the string and the string begins at byte 4.

TRACEFMT Unformatted Trace Buffer

The trace formatter (TRACEFMT) is able to save the unformatted STDA buffer for formatting at a later date. The format of this buffer is as follows:

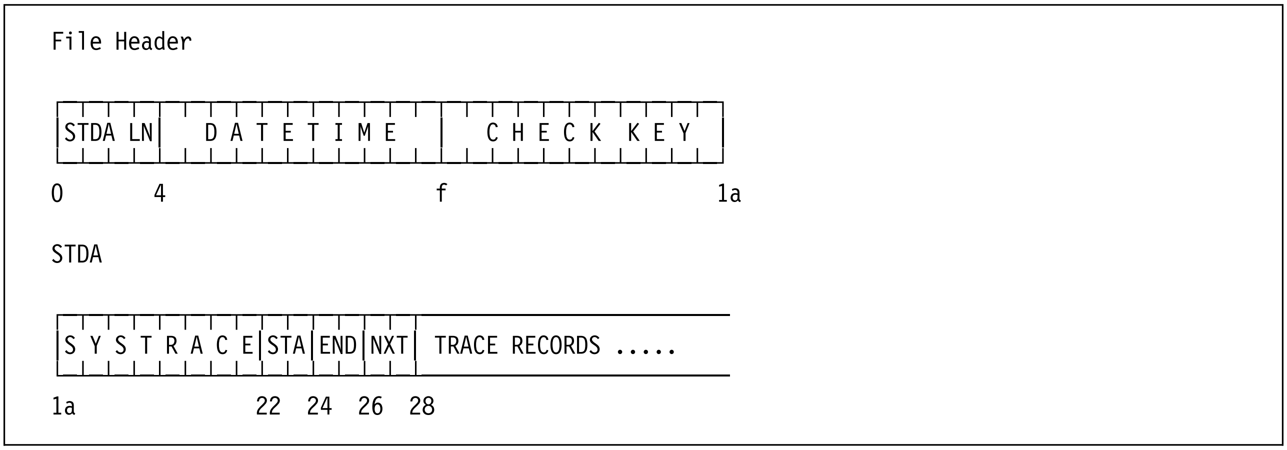


Figure 3. Unformatted Trace Buffer

Remarks

STDA LN
ULONG length of the STDA read by DosGetSTDA. Length is 1 greater than the STDA end offset.

DATETIME
A DATETIME structure returned by DosGetDateTime when this file buffer is created.

CHECK KEY
The DATETIME filed exclusively ORed with the string constant ″TRCFMTBUFF$″.

STDA
The STDA returned by DosGetSTDA.

Note: DosGetSTDA resets the internal start, end and next offsets after the STDA has been read. This allows trace formatting programs to detect an empty buffer.

For GA OS/2 2.x and OS/2 3.x the default start offset is 0x000e.

After FixPak 91 (OS/2 2.11) and FixPak 8 (OS/2 3.) the default start offset is 0x001e.

2.2.1.1 New STDA Format

From FixPak 8 (OS/2 3.0) and FixPak 91 (OS/2 2.11) the system trace was enhanced to provide improved timestamp information. Each trace records is timestamped in hours, minutes, seconds and 1/100 seconds. The trace logging start and stop times are also logged and displayed by the TRACEFMT command.

The spare bytes between the end of the STDA header and first trace record have been reserved for storing trace start and stop times. These are of the following format:

```
STDA

 ──┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬──────────────────
   │y│y│m│d│h│n│s│c│Y│Y│M│D│H│N│S│C│TRACE RECORDS ....
 ──┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴──────────────────
  28              30              38
```

Figure 4. STDA Spare Bytes

Where:

yymdhnsc is the TRACE ON date and time in years, month day, hours, seconds and 100th seconds.

YYMDHNSC is the TRACE OFF date and time in years, month day, hours, seconds and 100th seconds.

2.3 DosForceSystemDump (Force a System Stand-Alone Dump)

DosForceSystemDump allows an application to initiate a stand-alone system dump.

Syntax

```
    APIRET APIENTRY DosForceSystemDump(ULONG reserved);
32-bit code Example using CSet/2
```

Parameters

reserved
Reserved doubleword field that is set to 0L.

Returns

There is no return from this API.

Remarks

The system is halted abruptly and a stand-alone dump is initiated. After the stand-alone dump process has completed the system must be re-booted.

No shut down activity is performed when this API is called. File system buffers are not written to disk, cache is not flushed and files are not closed, *data loss may result*.

DosForceSystemDump is equivalent to using the Ctrl-Alt-Numlock-Numlock key sequence.

C Language prototype definitions for the DosForceSystemDump API may be found under 2.9, "RAS API Prototypes" on page 37.

To format a system dump, see the *OS/2 Debugging Library, Volume 2*.

For related information see:

- 1.10, "TRAPDUMP" on page 5.
- CREATEDD command in the OS/2 Command Reference.

2.4 DosDumpProcess (Enable/Disable ProcessDump)

DosDumpProcess allows an application:

- To enable or disable dynamically the process dump facility.
- To force a process dump for a given process.

The default setting is for process dump to be disabled unless overridden by the DUMPPROCESS CONFIG.SYS command in 1.6, "SUPPRESSPOPUPS" on page 4.

Syntax

```
APIRET APIENTRY DosDumpProcess(ULONG Flag, ULONG Drive, PID pid);
                32-bit code Example using CSet/2
```

Parameters

Flag
Doubleword field that may take one of the following values:

- (DDP_DISABLEPROCDUMP 0x00000000L)

  Disable process dumps.

- (DDP_ENABLEPROCDUMP 0x00000001L)

  Enable process dumps to be taken to a file in the root directory of a drive specified by the *Drive* parameter.

- (DDP_PERFORMPROCDUMP 0x00000002L)

Drive
Doubleword containing the ASCII value of the drive letter to which the PDUMP.nnn dump files will be written when DDP_ENABLEPROCDUMP is specified. For DDP_DISABLEPROCDUMP this parameter is ignored.

pid
Doubleword containing the process Id of the process to be dumped.

This option is valid only with DDP_PERFORMPROCDUMP. If zero is specified for PiD then the current process is dumped.

Returns.

Return Code

DosDumpProcess returns the following values:

0	NO_ERROR
87	ERROR_INVALID_PARAMETER
303	ERROR_INVALID_PROCID

Remarks

When process dump is enabled a dump file is written whenever a ring 3 process traps. The file takes the name *PDUMP.nnn* where *nnn* is incremented sequentially (staring from 000) for each successive dump.

The directory to which PDUMP.nnn will be written is always the root directory of *Drive*.

C Language prototype definitions for the DosDumpProcess may be found in 2.9, “RAS API Prototypes” on page 37.

The content of a process dump comprise register information at time of trap, system control blocks (TCB, TSD, PTDA, MTE, SMTE, OTE, VMAR, VMOB and LTD) that describe the state of the process at the time of error, ring 0 and ring 3 stack data for the trapping process.

See the process Dump Formatter section of the Dump Formatter User Guide for information on formatting Process Dumps.

Note: DDP_PERFORMPROCDUMP is not available in some early releases of OS/2 V2.11.

2.5 DosSuppressPopUps (Suppress Trap Exception Pop-Up Messages)

DosSuppressPopUps allows an application to enable or disable dynamically trap exception pop-up suppression and to specify the drive where the pop-up suppression log will be recorded.

The default setting is for disabled pop-up suppression unless overridden by the SUPPRESSPOPUPS CONFIG.SYS command in 1.6, “SUPPRESSPOPUPS” on page 4.

Syntax

```
APIRET APIENTRY DosSuppressPopUps(ULONG Flag, ULONG Drive);
                  32-bit code Example using CSet/2
```

Parameters

Flag
Doubleword field that may take one of the following values:

- (SPU_DISABLESUPPRESSION 0x00000000L)

  Disable pop-up suppression.

- (SPU_ENABLESUPPRESSION 0x00000001L)

  Enable pop-up suppression and pop-up logging to file POPUPLOG.OS2 on the drive specified by the *Drive* parameter.

Drive
Doubleword containing the ASCII value of the drive letter to which the POPUPLOG.OS2 log file will be written when SPU_ENABLESUPPRESSION is specified. With SPU_DISABLESUPPRESSION, *Drive* is ignored.

Returns

Return Code.

DosSuppressPopups returns the following values:

0	NO_ERROR
87	ERROR_INVALID_PARAMETER

Remarks

The directory to which POPUPLOG.OS2 will be written is always the root directory of *Drive*.

A prototype definition of DosSuppressPopUps may be found in 2.9, “RAS API Prototypes” on page 37.

See also the DosError API in the *OS/2 Control Program Programming Reference.*

2.6 DosQueryRASInfo (Query RAS Information)

DosQueryRASInfo returns information about active trace event recording and System Logging facility from the Global Information segment (InfoSegGDT) dump.

Syntax

```
APIRET  APIENTRY DosQueryRASInfo(ULONG Index, PPVOID Addr);
                  32-bit code Example using CSet/2
```

Parameters

Index
Doubleword field that may take one of the following values:

- (SPU_SIS_MEC_TABLE 0x00000001L)

Return the address of the table of actively traced major event codes in the InfoSegGDT. The table is 32 bytes long and each bit represents each major event code from 0 to 255.

- (SIS_SYS_LOG 0x00000002L)

  Return the address of the SYSLOG status word from InfoSegGDT. The status may contain a combination of:

  - (LF_LOGENABLE 0x0001) Logging enabled
  - (LF_LOGAVAILABLE 0x0002) Logging available

Returns

Return Code.

DosQueryRASInfo returns the following values:

0	NO_ERROR
5	ERROR_ACCESS_DENIED
87	ERROR_INVALID_PARAMETER

Remarks

For related information see:

- Logging facility
- The OS/2 Trace facility

2.7 16-Bit Error Logging APIs for IBM OS/2 Version 2.1

This section describes the ″Logging Facility for OS/2 2.1″. This comprises a set of three APIs, the logging deamon (LOG.SYS) and the log formatter (SYSLOG).

Both the Logging Deamon and Log Formatter are described in the *OS/2 Command Reference Manual* (see LOG.SYS under DEVICE statement of CONFIG.SYS and the SYSLOG command.

Note: C Language prototype definitions for the error logging APIs may be found under 2.9, “RAS API Prototypes” on page 37.

The following topics are described in this section:

- 2.7.1, “Dynamic vs. Static Error Log Record ID Registration” on page 19.
- 2.7.2, “DosLogRegister” on page 19.
- 2.7.3, “DosLogEntry” on page 21.
- 2.7.4, “DosLogRead” on page 23.
- 2.7.5, “Error Log Entry Formatting DLL Routines” on page 24.

2.7.1 Dynamic vs. Static Error Log Record ID Registration

OS/2 2.0 users of the DosLogEntry API will not need to use the DosLogRegister API. The DosLogRegister API is only maintained on OS/2 2.0 to support existing OS/2 1.3 programs that did need to use the API.

The OS/2 2.0 Version of the DosLogRegister API will always return a default Error Log record ID. It will accept a format template string as an input, but it will do nothing with the string since format template strings will not be saved within the OS/2 2.0 Version of the Error Log file.

The OS/2 2.0 Version of the DosLogEntry API will behave similarly to the OS/2 1.3 Version of the API. Since the OS/2 2.0 Version of the system Error Logging facility no longer supports the saving of format template strings within the Error Log file, it is necessary to provide a method by which DosLogEntry callers can associate their Error Log entry with a formatting (.DLL) routine. The OS/2 2.0 Version of the DosLogEntry API will make a special interpretation of the Originator Name field within the packet header. It will be assumed that this name field (if not NULL) contains the name of a Error Log formatting .DLL module.

2.7.2 DosLogRegister

There are two major differences between the OS/2 2.0 Version of DosLogRegister and the 1.3 version of the API:

- DosLogRegister no longer supports dynamic registration of Error Log record IDs. Instead, the API always returns a single default value.
- DosLogRegister no longer supports entry format template registration. While the API still accepts a format template as part of its input data packet, the format template will not be acted upon in any way.

DosLogRegister continues to support the existing alert notification registration function.

The description of the OS/2 2.0 Version of the DosLogRegister API follows:

Syntax

```
APIRET16 APIENTRY16 DosLogRegister((PUSHORT) LogHandle,
                                   (PVOID) LogRegList,
                                   (PUSHORT) RequestID)

                 32-bit code Example using CSet/2
```

Parameters

LogHandle
The address of the word in which the system will return the handle of a named pipe that will be transparently used in subsequent DosLogRead calls.

LogRegList
The address of the log registry buffer.

RequestID
The address of the word that the system will fill in with a default Error Log record ID (if the ′Error Log record ID′ field in the log registry buffer is set by the caller to -1)

Returns

Return code

DosLogRegister returns the following values:

0 Success

non-zero
Failure

Possible reasons for failure are:

- Facility unavailable
- Record ID in use
- Registration failed (general failure)
- Invalid ID
- Too many open files
- Too many semaphores
- Semaphore not found
- User semaphore limit reached
- Request timed out without satisfaction
- Error Log buffer temporarily full

Remarks

Length of the registration data	2
Reserved	2
Error Log record ID	2
Offset to the format template layout field	2
Semaphore name string	variable length
Format template layout	variable length

Figure 5. Log Registry Buffer Format Description

Where:

Length of the registration data
Is the total number of bytes in the current Log Registry Buffer (this length includes the two byte length field itself)

Reserved
Is a two byte reserved field

Error Log record ID
Contains the Error Log record ID that caller wishes to be registered for. If the field is set to 0xFFFF (-1), then a ″default″ record ID is returned in the word pointed to by the ′RequestID′ parameter. This field can be used to specify an alert notification record ID (that is, the caller wishes to be alerted whenever an Error Log Entry containing this record ID is logged).

Offset to the format template layout field
Is the offset within the Log Registry Buffer to the start of the format template layout area.

Semaphore name string
Is the name of a system semaphore, created with the nonexclusive option, that will be used to alert the caller′s process when an Error Log entry containing the specified ′Error Log record ID′ is logged. The name string is an ASCIIZ string.

Format template layout
Is an area within the Log Registry Buffer that contains the formatting structure information that is placed within the 1.3 Error Log file. This area is not used in the OS/2 2.0 Version of the DosLogRegister call. However, the ′length of the registration data′ field should reflect the size of this area.

In order to successfully resolve DosLogRegister function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGREGISTER=DOSCALL1.195
```

2.7.3 DosLogEntry

There are two major differences between the OS/2 2.0 Version of DosLogEntry and the 1.3 Version of the API:

- Since the DosLogRegister API will only return a ″default″ Error Log record ID to its caller, the DosLogEntry caller must override this ″default″ record with the appropriately statically allocated record ID if the caller wishes to see the ″correct″ record ID in the Error Log record.
- Since there is no explicit ″Error Log record formatting DLL module name″ field in the DosLogEntry log data packet, the API will attempt to interpret the ′Originator Name′ field in the packet′s header portion as a formatting DLL module name.

The description of the OS/2 2.0 Version of the DosLogEntry API is as follows:

Syntax

```
      APIRET16 APIENTRY16 DosLogEntry((USHORT) Function,
                                      (PVOID) LogData)

                         32-bit code Example using CSet/2
```

Parameters

Function
This specifies the type of log entry as follows:

0H	Reserved
1H	Error Logging

2H-FFFFH Reserved

LogData
This is the address of the log data buffer that contains one or more variable length log packets.

Returns

Return Code

DosLogEntry returns the following values:

0 Success

non-zero
Failure

Possible reasons for failure:

Invalid function

Facility unavailable

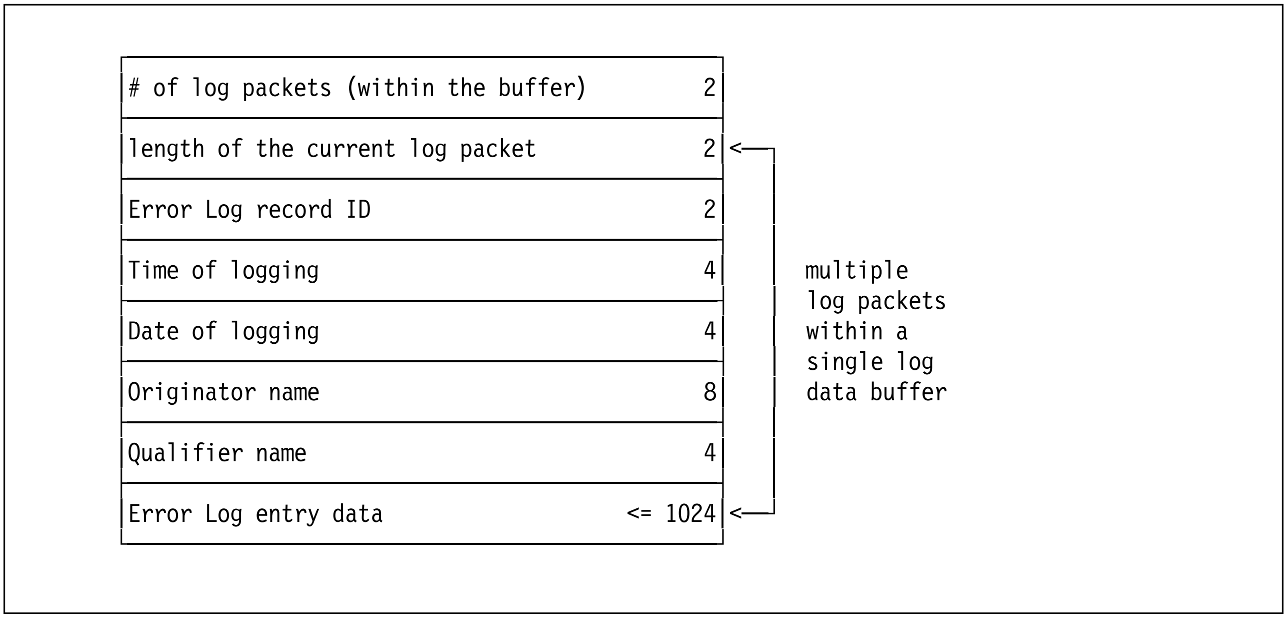
Facility suspended

Error Log buffer temporarily full

Remarks

Error Log Data Buffer format description:

Multiple log packets can be included within a single log data buffer. In the following diagram, the size of each field is indicated in bytes:



# of log packets (within the buffer)	2
length of the current log packet	2
Error Log record ID	2
Time of logging	4
Date of logging	4
Originator name	8
Qualifier name	4
Error Log entry data	<= 1024

Figure 6. Error Log Data Buffer Format Description

Where:

# of log packets
Is the number of separate packets contained within the user's buffer

Length of the current log packet
Is the number of bytes in the current log packet within the user′s log data buffer (this length includes the length of all the log packet control fields and the size of the Error Log entry data).

Error Log record ID
Is the record ID for the current Error Log entry (ID registration will be statically registered by the OS/2 development organization). The caller may pass in the ″default″ Error Log record ID that is returned by the DosLogRegister API.

Time of logging
Is filled in by the system Error Logging facility)

Date of logging
Is filled in by the system Error Logging facility

Originator name
Is a primary name field that is provided by the caller

Qualifier name
is a secondary name field that is provided by the caller

Error Log entry data
Is an optional variable length set of data that can be supplied by the caller (the format of the data is under the control of the caller).

In order to successfully resolve DosLogEntry function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGENTRY=DOSCALL1.193
```

2.7.4 DosLogRead

The description of the OS/2 2.0 Version of the DosLogRead API follows:

```
APIRET16 APIENTRY16 DosLogRead((USHORT) LogHandle,
                               (USHORT) Length,
                               (PVOID) LogBuffer,
                               (PUSHORT) ReadSize)
```

32-bit code Example using CSet/2

Parameters

LogHandle
This is the named pipe handle returned by **DosLogRegister()**

Length
This is the length (in words) of the caller′s log buffer

LogBuffer
This is the address of the caller′s buffer, into which the system Error Logging facility will place a single Error Log entry packet (formatted in the manner of the 16-bit DosLogEntry API).

ReadSize
This is the address of a word, into which the system Error Logging facility will place the number of bytes that it wrote into the caller′s log buffer. If a zero is returned here, then there was no Error Log packet to return.

Returns

Return code

DosLogRead returns the following values:

0 Indicating success.

non-zero
Indicating error

Possible reasons for failure:

- Invalid log handle
- Facility unavailable
- Buffer too small

In order to successfully resolve DosLogRead function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGREAD=DOSCALL1.196
```

DosLogRead returns Error Log entries that are formatted in the manner of the 16-bit DosLogEntry API.

2.7.5 Error Log Entry Formatting DLL Routines

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the DosLoadModule API to create a run-time link to the specified formatting DLL module. It uses the DosFreeModule API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of *colliding* with another DLL module of the same name, it is suggested that a formatting DLL module be labelled with a name that adheres to the following standard form:

```
    ELGxxxxx.DLL    (where ″xxxxx″ corresponds to the Error
                       Log record ID (in decimal) of any one
                       of the types of records that the formatting
                       routine is designed to handle)
```

For example,

```
″ELG00127.DLL″ is a standardized name for a formatting DDL
          module that recognizes (among other things)
          Log records with ID of 127 (decimal)
```

This standard naming convention is suggested because it is assumed that the Error Log records of any one ID will only be recognized by a single formatting routine. Therefore the use of the ″xxxxx″ suffix (based on record ID) should assure uniqueness for the formatting module name.

The static Error Log record ID registration mechanism that is enforced by the OS/2 development organization will attempt to keep a list not only of the Error Log record IDs in use, but also the names of the formatting DLL modules that correspond to each record ID. This will also help to reduce the possibility of formatting DLL module names *colliding*.

In addition to its single formatting routine, each formatting DLL module must contain a global variable named ″ELOG_FORMAT″. For OS/2 2.0, this exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it attempts to access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable will in future releases be used as a revision level for the SYSLOG/formatting DLL module interface specification.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel. The SYSLOG utility will be written to use the *ordinal* version of the DosGetProcAddr API.

Error Log record formatting DLL routines must be written as 32-bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```
ULONG ELGxxxxx((PVOID) Log_Record, (PVOID) String_Buffer,
               (ULONG) Buffer_Length, (PULONG) String_Length)
```

Parameters

Log_Record

A linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled ″Error Log File Entry Format″, except that the linear pointer points to the ″TOT_LENGTH″ field (since the ″PREV_PTR″ and ″PREV_SIZE″ fields are of no interest to a formatting routine).

String_Buffer

Is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string ″line″ within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

Buffer_Length
Is a 32-bit integer that contains the maximum size of the ′String_Buffer′.

String_Length
Is a pointer to a 32-bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in the ′String_Buffer′.

Returns

ELGxxxxx returns the following:

0 Indicating success

-1 Indicates insufficient space in the ′String_Buffer′, positive values indicate formatting routine errors.

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the ′String_Buffer′, then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller′s output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger ′String_Buffer′.

SYSLOG will contain logic to format the standard SNA Generic Alert entry (that is, Error Log record ID of 2). This is necessary since most of the existing Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record ID of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

2.8 32-Bit Error Logging APIs for IBM OS/2 Version 2.1 and 3.0

This section describes the ″Logging Facility for OS/2 2.1 and 3.0″. This comprises a set of four APIs, a DevHlp function, the logging deamon (LOGDAEM.EXE), the logging device driver (LOG.SYS) and the log formatter (SYSLOG).

The Logging Deamon, Device Driver and Log Formatter are described in the OS/2 Command Reference - see LOG.SYS under the DEVICE statement of CONFIG.SYS and the SYSLOG command.

Note: C Language prototype definitions for the Error Logging APIs may be found in 2.9, "RAS API Prototypes" on page 37.

The following topics are described in this section:

- 2.8.1, "LogOpen"
- 2.8.2, "LogClose" on page 28
- 2.8.3, "LogAddEntries" on page 28
- 2.8.4, "LogGetEntries" on page 32
- 2.8.5, "32-Bit Error Log Entry Formatting DLL Routines" on page 33
- 2.8.6, "DevHlp_LogEntry Device Driver Interface" on page 36

The set of four 32-bit logging APIs provide equivalent functionallity to the three 16-bit logging APIs discussed in the previous section. They may be used as a complete replacement to the 16-bit set.

2.8.1 LogOpen

LogOpen is a 32-bit system Error Logging facility high level API. It is used to open a connection to the facility (through the System Logging Service device driver).

The description of the LogOpen API call follows:

Syntax

```
APIRET APIENTRY LogOpen(PHFILE phf);
```

Parameters

phf
This points to a file handle holder that on return will hold an open file handle

Returns

Return code

LogOpen returns the following values:

0 Success

non-zero
Facility not available

Remarks

The file handle that is returned by the LogOpen API is required in all subsequent high level system Error Logging facility API calls.

In order to resolve successfully LogOpen function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogOpen=DOSCALL1.430
```

2.8.2 LogClose

LogClose is a 32-bit system Error Logging facility high level API. It is used to close a connection to the facility.

The description of the LogClose API call follows:

Syntax

```
        APIRET APIENTRY LogClose(HFILE hf);
```

Parameters

hf Is the file handle returned by LogOpen()

Returns

Return code

LogClose returns the following values:

0 Success

non-zero
Failure, possible reason: facility not open

Remarks

In order to resolve successfully LogClose function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogClose=DOSCALL1.431
```

2.8.3 LogAddEntries

LogAddEntries is a 32-bit system Error Logging facility high level API. It is used to allow application processes to add Error Log entries to the internal Error Log buffer that is maintained by the System Logging Service device driver.

The description of the LogAddEntries API call follows:

Syntax

```
        APIRET APIENTRY LogAddEntries(HFILE hf, ULONG service,
                                      PVOID log_data_address);
```

Parameters

hf Is the file handle returned by LogOpen()

service

Specifies the class of logging facility:

0x0 Reserved

0x1 Error Logging

0x2 - 0xffff Reserved

log_data_address

Is the address of a buffer that contains a variable length Error Log entry. The first word of the buffer contains the number of packets in the Error Log entry

Returns

Return code

LogAddEntries return the following values:

0 Success

non-zero

Failure

Possible reasons for failure are:

Invalid log type

Facility unavailable

Facility suspended

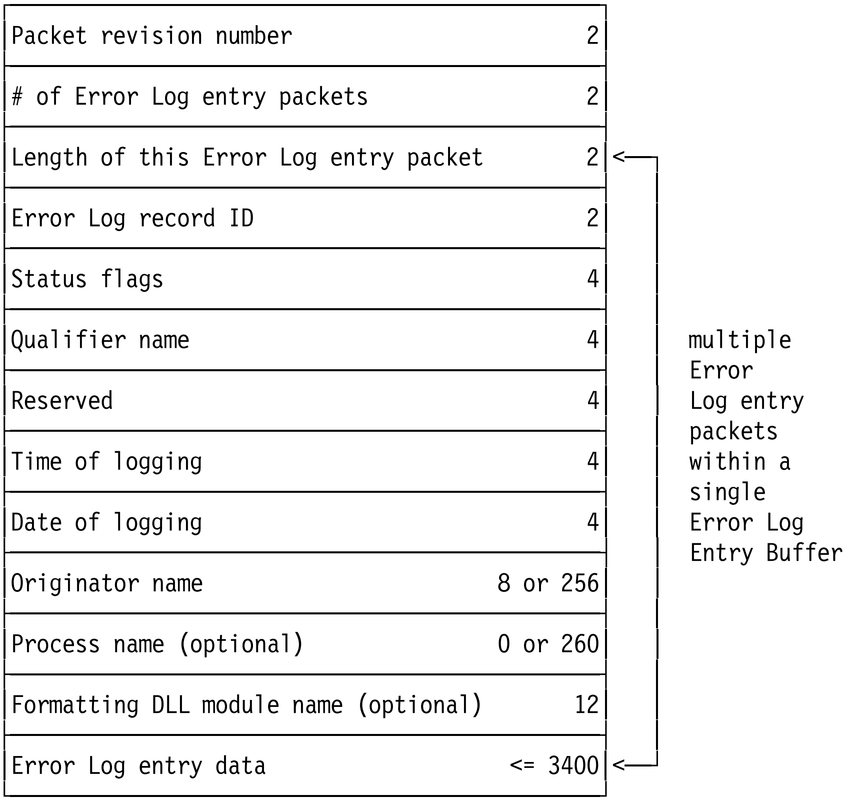
Facility not open

Error Log buffer temporarily full

Remarks

Error Log Entry Buffer format description:

Multiple Error Log packets can be included within a single Error Log entry buffer. If multiple packets are included within a single buffer, each individual packet should be aligned on a double word boundary. In the following diagram, the size of each field is indicated in bytes:



Packet revision number	2
# of Error Log entry packets	2
Length of this Error Log entry packet	2
Error Log record ID	2
Status flags	4
Qualifier name	4
Reserved	4
Time of logging	4
Date of logging	4
Originator name	8 or 256
Process name (optional)	0 or 260
Formatting DLL module name (optional)	12
Error Log entry data	<= 3400

multiple Error Log entry packets within a single Error Log Entry Buffer

Figure 7. Error Log Entry Buffer Format Description

Where

Packet revision number
Is an integer value that can be used to distinguish error logging packets that are intended to be handled by different revisions of the LogAddEntries API. For the initial version of the API, this field should be set to a value of 1. This field is included in the packet to support future backward compatibility.

# of Error Log entry packets
Is the number of separate packets contained within the user′s buffer.

Length of this Error Log entry packet
Is the number of bytes in the current Error Log entry packet within the user′s Error Log Entry Buffer (this length includes the length of all the Error Log entry packet control fields and the size of the Error Log entry text). To support efficient logging execution, this length should be a multiple of 4 bytes (i.e. if necessary the user should pad the Error Log entry packet).

Error Log record ID
Is the record ID for the current Error Log entry (ID registration will be statically registered by the OS/2 development organization).

Status flags
Is a two byte flag holder that contains three single bit flags:

(BIT 0) is used to indicate whether the current Error Log entry packet contains space in which the Error Logging facility can place a long process name (″on″ indicates YES, ″off″ indicates NO);

(BIT 1) is used to indicate whether the current Error Log entry packet contains an 8 byte originator name or a 256 byte originator name (″on″ indicates a 256 byte originator name, ″off″ indicates an 8 byte originator name);

(BIT 2) is used to indicate that the caller has placed time and date values in the Error Log entry packet and does not wish to have those values modified during the logging process (″on″ indicates that the Error Log entry packet already contains time and date values, ″off″ indicates the packet does not already contain time and date values);

All the other 29 bits in ′status flags′ are considered reserved at this time and will be zeroed by the **LogAddEntries** API.

Qualifier name
Is a secondary name field that is provided by the caller

Reserved
Is a four byte reserved field

Time of logging
Is filled in by the system Error Logging facility (unless BIT 2 of the ′status flags field is ″on″, indicating that the caller has preset a time value).

Date of logging
Is filled in by the system Error Logging facility (unless BIT 2 of the ′status flags field is ″on″, indicating that the caller has preset a date value);

Originator name
Is a primary name field that is provided by the caller.

Process name
Is an optional long process name field that will be filled in by the Error Logging facility if the field is provided by the caller in the Error Log entry packet.

Formatting DLL module name
Is the optional name of a DLL module that houses a formatting routine that recognizes this type of Error Log entry and can format it for display by the SYSLOG utility. The name is specified as an ASCIIZ string that can be up to eight characters in length. If no module name is specified in this field, then SYSLOG will display the data portion of the Error Log entry as a hexadecimal dump.

Error Log entry data
Is an optional variable length set of data that can be supplied by the caller (the format of the string is under the control of the caller).

The format and function of the LogAddEntries API call is very similar to that of the 16-bit DosLogEntry call. There are several functional differences from the DosLogEntry call:

- The user-supplied error log entry Record ID will now be a statically allocated value rather than a dynamically allocated value.
- The maximum size of the originator name field in the caller′s packet has been increased from 8 bytes to 256 bytes. The caller can specify whether

the packet contains an 8 byte originator name field or a 256 byte originator name field.

- The maximum size of the variable length data portion within the caller′s packet has been increased from 1024 bytes to 3400 bytes
- The order of the fields within the Error Log entry has been slightly rearranged to support the creation of smaller internal control messages.

In order to resolve successfully LogAddEntries function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogAddEntries=DOSCALL1.432
```

2.8.4 LogGetEntries

LogGetEntries is a 32-bit system Error Logging facility high level API. It is used to allow application processes to obtain Error Log entries from the internal Error Log buffer that is maintained by the System Logging Service device driver.

LogGetEntries description:

Syntax

```
        APIRET APIENTRY LogGetEntries(HFILE hf, ULONG service,
                                      ULONG type, PVOID buffer,
                                      ULONG buffer_length);
```

Parameters

hf is the file handle returned by LogOpen()

service
Specifies the class of logging facility:

0x0	Reserved
0x1	Error Logging
0x2 - 0xffff	Reserved

type
Specifies the class of internal logging buffer to read:

0x0	Reserved
0x1	Buffer that contains all logged entries
0x2	Buffer that only contains entries that were logged by device drivers
0x3 - 0xffff	Reserved

buffer
Is a pointer to a buffer that will receive entries copied from the internal logging service buffer (if the caller′s buffer is too small to fit all the current entries in the device driver Error Log buffer, then on return the first double word of the buffer will be set to the size of the Error Log buffer, expressed as a number of bytes).

buffer_length
Thi is the length of the caller′s buffer.

Returns

Return Code

LogGetEntries returns the following values:

0 Success

non-zero
Failure

Possible reasons for failure:

- Facility not open
- Facility unavailable
- Buffer too small

Remarks

The format and function of the LogGetEntries API call is very similar to that of the 16-bit DosLogGetBuffer call.

In a similar fashion to the DosLogGetBuffer API, if the caller′s buffer is too small to fit all the current entries in the device driver Error Log buffer, an error return code is set indicating this problem, and no Error Log entries are placed in the caller′s buffer. In this case, the first doubleword of the buffer is set to the size of the Error Log buffer (expressed as a number of bytes). If this error occurs, the caller should repeat the LogGetEntries call with a larger buffer.

In order to resolve successfully LogGetEntries function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogGetEntries=DOSCALL1.433
```

2.8.5 32-Bit Error Log Entry Formatting DLL Routines

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the DosLoadModule API to create a run-time link to the specified formatting DLL module. It also uses the DosFreeModule API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of *colliding* with another DLL module of the same name, it is suggested that a formatting DLL module be labeled with a name that adheres to the following standard form:

```
ELGxxxxx.DLL    (where ″xxxxx″ corresponds to the Error
                   Log record ID (in ecimal) of any one
                   of the types of records that the formatting
                   routine is designed to handle)
```

For example,

```
″ELG00127.DLL″ is a standardized name for a formatting
 DDL module that recognizes (among other things) Error
 Log records with ID of 127 (decimal)
```

This standard naming convention is suggested because it is assumed that the Error Log records of any one ID will only be recognized by a single formatting routine. Therefore the use of the ″xxxxx″ suffix (based on record ID) should assure uniqueness for the formatting module name.

The static Error Log record ID registration mechanism that is enforced by the OS/2 RAS development group will attempt to keep a list not only of the Error Log record IDs in use, but also the names of the formatting DLL modules that correspond to each record ID. This may also help to reduce the possibility of formatting DLL module names *colliding*.

In addition to its single formatting routine, each formatting DLL module must contain a global variable named ″ELOG_FORMAT″. This exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it will attempt to access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable may in future releases be used a sort of revision level for the SYSLOG/formatting DLL module interface specification. That is why it will initially be forced to a value of 1.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel.

Error Log record formatting DLL routines must be written as 32-bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```
APIRET APIENTRY ELGxxxxx(PVOID Log_Record,
                         PVOID String_Buffer,
                         ULONG Buffer_Length,
                         PULONG String_Length);
```

Parameters

Log_Record
Is a linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled ″Error Log File Entry Format″, except that the linear pointer points to the ″TOT_LENGTH″ field (since the ″PREV_PTR″ and ″PREV_SIZE″ fields are of no interest to a formatting routine).

String_Buffer
Is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string ″line″ within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

Buffer_Length
Is a 32-bit integer that contains the maximum size of the the ′String_Buffer′.

String_Length
Is a pointer to a 32-bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in ′String_Buffer′.

Returns

ELGxxxxx returns the following:

0 Indicating success.

-1 Indicates insufficient space in ′String_Buffer′ positive values indicate formatting routine errors.

Remarks

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the ′String_Buffer′, then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller′s output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger ′String_Buffer′.

SYSLOG contains logic to format the standard SNA Generic Alert entry (For example, Error Log record ID of 2). This is necessary since most of the existing

Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record ID of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

2.8.6 DevHlp_LogEntry Device Driver Interface

DevHlp_LogEntry provides a device driver interface to the logging facility.

The description of the LogEntry DevHlp function follows:

```
Calling sequence -  LES   BX,log_data_address
                    MOV   CX,service
                    MOV   DL,DevHlp_LogEntry  /* LogEntry function
                                               code 0x3b */
                    CALL  [Device_Help]
```

Parameters

log_data_address
This is the address of a buffer that contains a variable length Error Log entry. (See the section on the LogAddEntries high level API for further details.) level API for further details.)

service
This is the class of logging facility:

0x0	Reserved
0x1	″Old-Style″ Error Logging call (″old″ 16-bit (DosLogEntry-style) data packet provided).
0x2 - 0x2f	Reserved for future use.
0x80 - 0x8f	Reserved for internal use by the System Logging Service device driver.
0x90	″New_Style″ Error Logging call (″new″ 32-bit (LogAddEntries-style) data packet provided).
0x91 - 0xffff	Reserved for future use.

Returns

Return code in AX:

0 Success

non-zero
Failure

Possible errors:

Invalid log type

Facility unavailable

Facility suspended

Remarks

When CX is set to 80H, DS:SI is set to point to the device driver header block of the System Logging Service device driver.

2.9 RAS API Prototypes

The following is a sample C language header file that contains sample prototype definitions for the RAS APIs.

```
/* definitions for DosDumpProcess */
#define DDP_DISABLEPROCDUMP     0x00000000L     /* disable process dumps */
#define DDP_ENABLEPROCDUMP      0x00000001L     /* enable process dumps */
#define DDP_PERFORMPROCDUMP     0x00000002L     /* perform process dump */

/* definitions for DosSuppressPopUps */
#define SPU_DISABLESUPPRESSION  0x00000000L     /* disable popup suppression */
#define SPU_ENABLESUPPRESSION   0x00000001L     /* enable popup suppression */

/* definitions for DosQueryRASInfo Index */
#define SIS_MMIOADDR            0
#define SIS_MEC_TABLE           1
#define SIS_SYS_LOG             2
#define LF_LOGENABLE    0x0001          /* Logging enabled */
#define LF_LOGAVAILABLE 0x0002          /* Logging available */

APIRET  APIENTRY        DosQueryRASInfo(ULONG Index, PPVOID Addr);

APIRET  APIENTRY        DosForceSystemDump(ULONG reserved);

APIRET  APIENTRY        DosDumpProcess(ULONG Flag, ULONG Drive, PID Pid);

APIRET  APIENTRY        DosSuppressPopUps(ULONG Flag, ULONG Drive);

APIRET16  APIENTRY16    DosSysTrace(USHORT Majorcode, USHORT Length,
                                    USHORT Minorcode, PCHAR pData);

APIRET16 APIENTRY16 DosGetSTDA(SEL, SHORT, SHORT );

/* 32-bit Logging Facility Function Prototypes                   */

/*--------------------------------------------*/
/* Logging Defines                            */
/*--------------------------------------------*/
#define ERRLOG_SERVICE        1L
#define ERRLOG_VERSION        1

/*--------------------------------------------*/
/* LogRecord status bits                      */
/*--------------------------------------------*/
#define LF_BIT_PROCNAME     0x0001L
#define LF_BIT_ORIGIN_256   0x0002L
#define LF_BIT_DATETIME     0x0004L
#define LF_BIT_SUSPEND      0x0008L
#define LF_BIT_RESUME       0x0010L
#define LF_BIT_REDIRECT     0x0020L
#define LF_BIT_GETSTATUS    0x0040L
#define LF_BIT_REGISTER     0x0080L
#define LF_BIT_REMOTE_FAIL  0x0100L
```

```
/*----------------------------------------------*/
/*    Log Entry Record Header for 2.X         */
/*    This is format used by 2.0 device       */
/*     drivers and callers of LogAddEntries   */
/*----------------------------------------------*/
typedef struct LogRecord
{
  USHORT   len ;                   /* this record length(includes len field)*/
  USHORT   rec_id ;                /* record id                             */
  ULONG    status ;                /* record status bits(see LF_BIT_)       */
  UCHAR    qualifier[4] ;          /* qualifier tag                         */
  ULONG    reserved ;
  ULONG    time ;                  /* hours minutes seconds hundreds        */
  ULONG    date ;                  /* day month (USHORT)year                */
  UCHAR    data[1] ;               /* begin of variable data that includes: */
                               /* Originator(256 bytes if LF_BIT_ORIGIN_256)*/
                               /* else 8 bytes long                         */
                               /* Processname(260 bytes) only if status     */
                               /* LF_BIT_PROCNMAME set                      */
                               /* FormatDLLName(12 bytes)                   */
                               /* Variable data                             */
} LOGRECORD ;
typedef LOGRECORD far *PLOGREC ;

/*------------------------------------------------*/
/* Format of buffer sent to LogAddEntries       */
/*------------------------------------------------*/
typedef struct LogEntryRec
{
  USHORT    version ;                            /* this version is 1       */
  USHORT    count   ;                /* number of log records in this buffer*/
  LOGRECORD logrec  ;                            /* repeated count times    */
} LOGENTRYREC ;
typedef LOGENTRYREC far *PLOGENTRYREC ;

/*----------------------------------------------------*/
/* Logging facility Function prototypes             */
/*----------------------------------------------------*/
APIRET APIENTRY LogOpen( PHFILE phf );

APIRET APIENTRY LogClose( HFILE hf );

APIRET APIENTRY  LogAddEntries( HFILE hf, ULONG ulService, PVOID pLogEntries );

APIRET APIENTRY LogGetEntries( HFILE hf, ULONG ulService, ULONG ulType,
                               PVOID pLogBuffer, ULONG ulBufferLen );

/* 16-bit Logging Facility Function Prototypes                   */


APIRET16 APIENTRY16     DosLogRegister(PUSHORT LogHandle,
                                       PVOID LogRegList,
                                       PUSHORT RequestID);


APIRET16 APIENTRY16     DosLogEntry(USHORT Function,
                                    PVOID LogData);

APIRET16 APIENTRY16     DosLogRead(USHORT LogHandle,
USHORT Length,
PVOID LogBuffer,
PUSHORT ReadSize);
```

Chapter 3. OS/2 System Control Block Reference

This chapter contains details of some of the more important system control blocks used in debugging.

Where major differences in format exist between ALLSTRICT and RETAIL, and between versions of OS/2 then each version of the control block is given. Otherwise only OS/2 Warp V3.0 ALLSTRICT Kernel versions of the control blocks are given and may be assumed to be applicable to also RETAIL and earlier versions of OS/2.

Attention

The information given in this chapter is for debugging purposes only. The layout of the control blocks may change from one release of OS/2 to the next. They are not to be considered a programming interface.

The following system components are included in this chapter and an overview is provided in the next section, 3.1, "Overview of Kernel Components and Interfaces" on page 43.

3.2, "Miscellaneous System Control Block Reference" on page 47
This section describes system structures that are common to all components. These include SAS and RMP.

3.3, "Semaphore Control Block Reference" on page 55
This section describes the control blocks used for RamSem, FSRamSem Ksem, SysSem, PM/GRE, 32-bit, and MuxWait Semaphores.

3.4, "Memory Management Control Block Reference" on page 69
This section describes the following control blocks used by memory management:

VMAL, VMOB, VMAR, VMCO, VMAT, VMAH, PF and VP

3.5, "Scheduler Thread and Process Control Block Reference" on page 89
This section describes the following control blocks used by thread and process management:

PTDA, TCB, TSD, GISEG, LISEG, PIB, TIB, EXENT and exception handler structures

3.6, "Loader Control Block Reference" on page 165
This section describes the following control blocks used by the system loader component:

MTE, SMTE, OTE, STE

3.7, "File System Block Reference" on page 172
This section describes the following control blocks used by the file system component:

SFT, MFT, FSC, RLR, VPB, DBP, CDS, BUF, Named and Anonymous Pipes

3.8, "I/O System Control Block Reference" on page 205
This section describes the structures that relate to low level I/O. These include: Request Packets, BIOS Parameters Blocks and Device Driver Headers, Virtual Device Driver Entry Points.

Format

The control block formatting conventions have been chosen to aid the user of the Kernel Debugger and Dump Formatter.

Each control block is presented in tabular form with five columns used as follows:

name
Field name, usually taken from the C header or MASM include file definition.

Off Offset from the beginning of the structure. The offset is of the form ***x.y*** where ***x*** is the signed hexadecimal byte offset from the beginning of the structure and ***y*** is the bit offset from the high-order bit of the byte.

Leng
Hexadecimal length of the field.

Type
The field type, for the purposes of displaying storage using the D command. The following values are used:

S	Complex structure. Choose display command to best suit your needs.
D	Double word. Use **DD** to format the field correctly.
W	Word. Use **DW** to format the field correctly.
B	Byte. Use **DB** to format the field correctly.
A	ASCII byte string. Use **DA** to format the field correctly.
blank	A blank value appears when a field does not begin or end on a byte boundary. In this case format the field from the previous field for which a type value is given. Such bit fields are presented in an order assuming this instruction is followed. *Attempts to display bit fields in other ways may lead to a great deal of confusion!*

Description
field description taken usually from the header or include file.

A null row is used to indicate an overlay definition of the same control block.

Flag fields are separately formatted in tabular form.

Where a flag field represents a bit mask, the mask is given in hexadecimal and is assumed to indicated that corresponding bits are set to be in effect. Exceptions are specifically notes in the description.

When the flag field take numerical values then they will be shown in either hexadecimal (prefixed with **0x**) or decimal depending on the C or MASM definitions.

3.1 Overview of Kernel Components and Interfaces

The OS2KRNL modules lies at the heart of OS/2; it is essentially the operating system.

The kernel comprises of a number of internal components, each responsible for a different aspect of running the system. It also has a number of interfaces that provide services to applications, device drivers and file systems.

These aspects are now considered in a little more detail and are summarized in the diagram shown in Figure 8 on page 46.

3.1.1 Kernel Components

Task management and the Scheduler

This is responsible for thread and process management. The functions performed include:

- Thread and process creation and termination
- Thread scheduling (priority and state management)
- Preparing threads for dispatching
- Blocking and running
- Implementing the thread and process related APIs

The scheduler′s principle control blocks are:

PTDA	Per Task Data Area
TCB	Thread Control Block
TSD	Thread Swappable Data
TSS	Task State Segment (H/W)

System Loader

This is responsible for load module management. The loader′s principle responsibilities include:

- Bringing modules into memory and performing fixups
- Managing modules resources
- Managing dynamic linking
- Tracking module references
- Deleting modules from memory
- Managing the discarding and swapping of module pages
- Implementing module related APIs

The loader′s principle control blocks are:

MTE	Module Table Entry
SMTE	Swappable Module Table Entry
OTE	Object Table Entry
STE	Segment Table Entry

Memory Management

Memory management is responsible for managing physical, virtual, and swapper memory. Its principle roles include:

- Allocation and assignment of physical pages of memory
- Allocation and assignment of virtual storage
- Managing the swapper
- Memory locking
- Implementing memory related APIs

The principle control blocks of memory management are:

VMAR	Virtual Memory Arena Record
VMOB	Virtual Memory Object Record
PF	Page Frame Structure
VP	Virtual Page Structure
PTE	Page Table Entry (H/W)

File System

The file system kernel component responsibilities include:

- Access to FAT formatted media.
- Interfacing with file system drivers for accessing non-FAT media
- Managing and tracking the status of all open files.
- Path management
- File sharing and serialization.
- Providing helper kernel services for FSDs.
- Implementation of all file system APIs.

The principle control blocks of the file system include:

MFT	Master File Table Entry
SFT	System File Table Entry
CDS	Current Directory Structure
FSC	File System Control Block

Device and I/O Management

This component is responsible for interfacing with physical device drivers. Its responsibilities include:

- Routing requests to PDDs from applications
- Managing interrupts
- Providing helper kernel services for PDDs

The principle control blocks for device management include:

IRQI	IRQ Information Array
DIRQ	Device IRQ Information

REQ PDD request Packet.

DEV PDD device header

Virtual Dos Machine
This component is responsible for providing the entire DOS Machine emulation. This has not been covered in this book, except for the Virtual Device Driver interface.

3.1.2 Kernel Interfaces

The Kernel provides the following external interfaces:

Application (R3/2) Interface
Application access kernel services via GDT call gates. These are called either directly from the application program or via the DOSCALL1.DLL module, where additional Ring 3 processing is required before calling the kernel. Some system interfaces are able to be implemented entirely within Ring 2/3. In these cases, DOSCALL1.DLL does not make any kernel calls.

The kernel interfaces are represented by a fictitious module called DOSCALLS.DLL.

File System Driver (FSD)
The FSDs run in ring 0 as separately loaded modules. They provid a set of interfaces to the kernel via the FSD_Hlp (File System Helper) calls.

Physical Device Driver (PDD)
The PDDs run in ring 0 as separately loaded modules. They provid a set of interfaces to the kernel via the Dev_Hlp (Device Helper) calls.

Virtual Device Driver (VDD)
The VDDs run in ring 0 as separately loaded modules. They provid a set of interfaces to the kernel via the VDD_Hlp (Virtual Device Driver Helper) calls.

Compatibility BIOS
The compatibility BIOS resides within the OS2LDR module. It provides a hardware implementation independent layer through which the kernel accesses the BIOS. The interface to the CBIOS from the kernel is provided by the Dos_Hlp (Dos Helper Services). These are not available for access by PDDs, VDDs or FSDs, however a limited set of Dos_Hlp calls are provided via the TESTCFG.SYS and OEMHLP$ device drivers.

Notes: OEMHLP$ is not a separately loaded module; it is resident within the OS2LDR module.

OS2LDR is responsible for loading the Kernel at system initialization time. It does not get involved with the loading of application programs, PDDs, VDDs for FSDs during normal running; that function is performed by the system loader component of the Kernel.

3.1.2.1 The OS/2 Kernel's Interfaces

OS/2 Kernel Interfaces

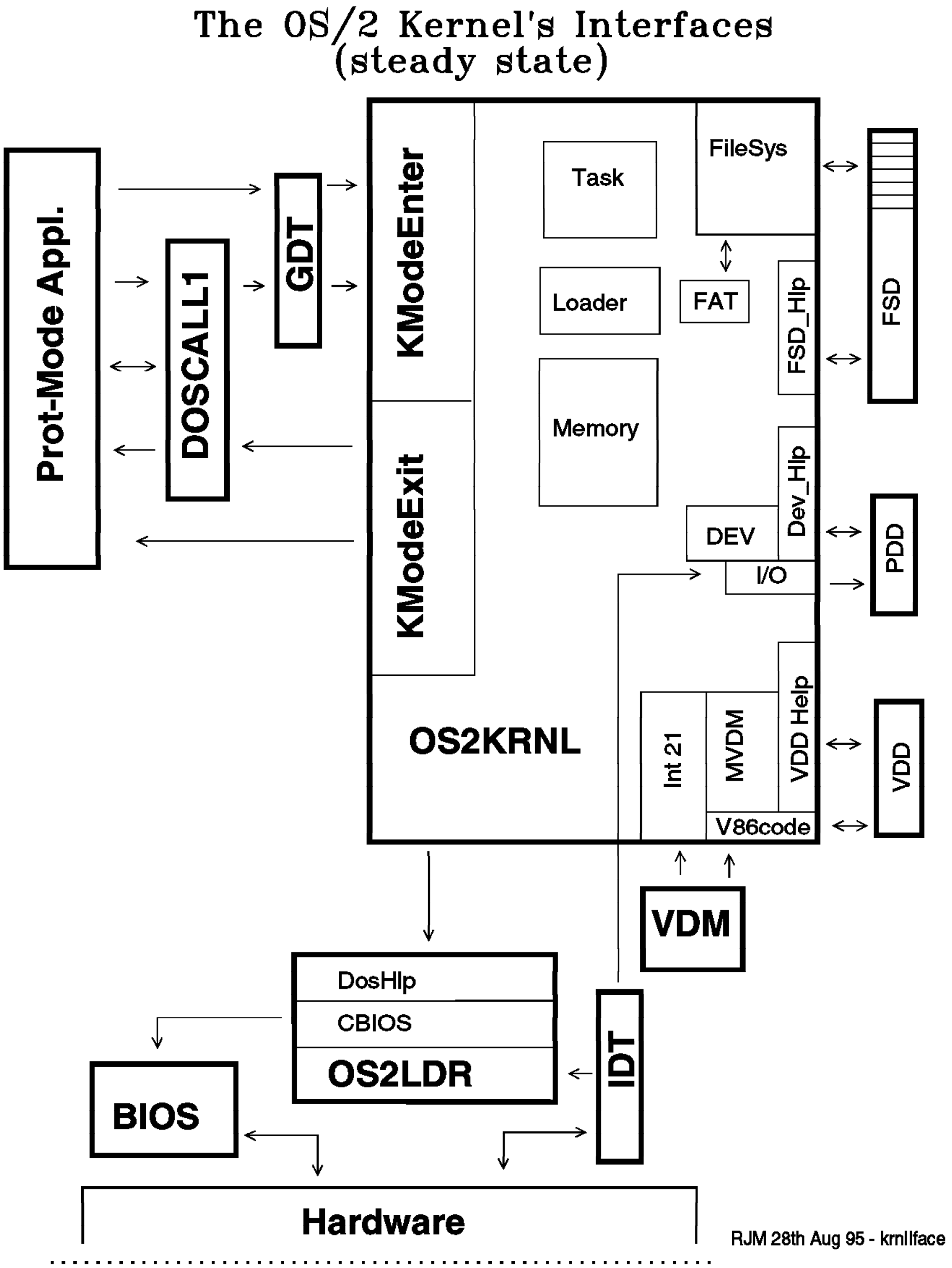


Figure 8. OS/2 Kernel Interfaces

3.2 Miscellaneous System Control Block Reference

The following control blocks are described in this section:

3.2.2, “System Anchor Segment (SAS) for OS/2 Warp V3.0” on page 49

3.2.3, “Record Management Package (RMP) for OS/2 Warp V3.0” on page 52

An overview of the miscellaneous system control blocks is as follows:

3.2.1 Miscellaneous System Diagrams

The following diagram illustrates the System Anchor Segment.

3.2.1.1 The System Anchor Segment

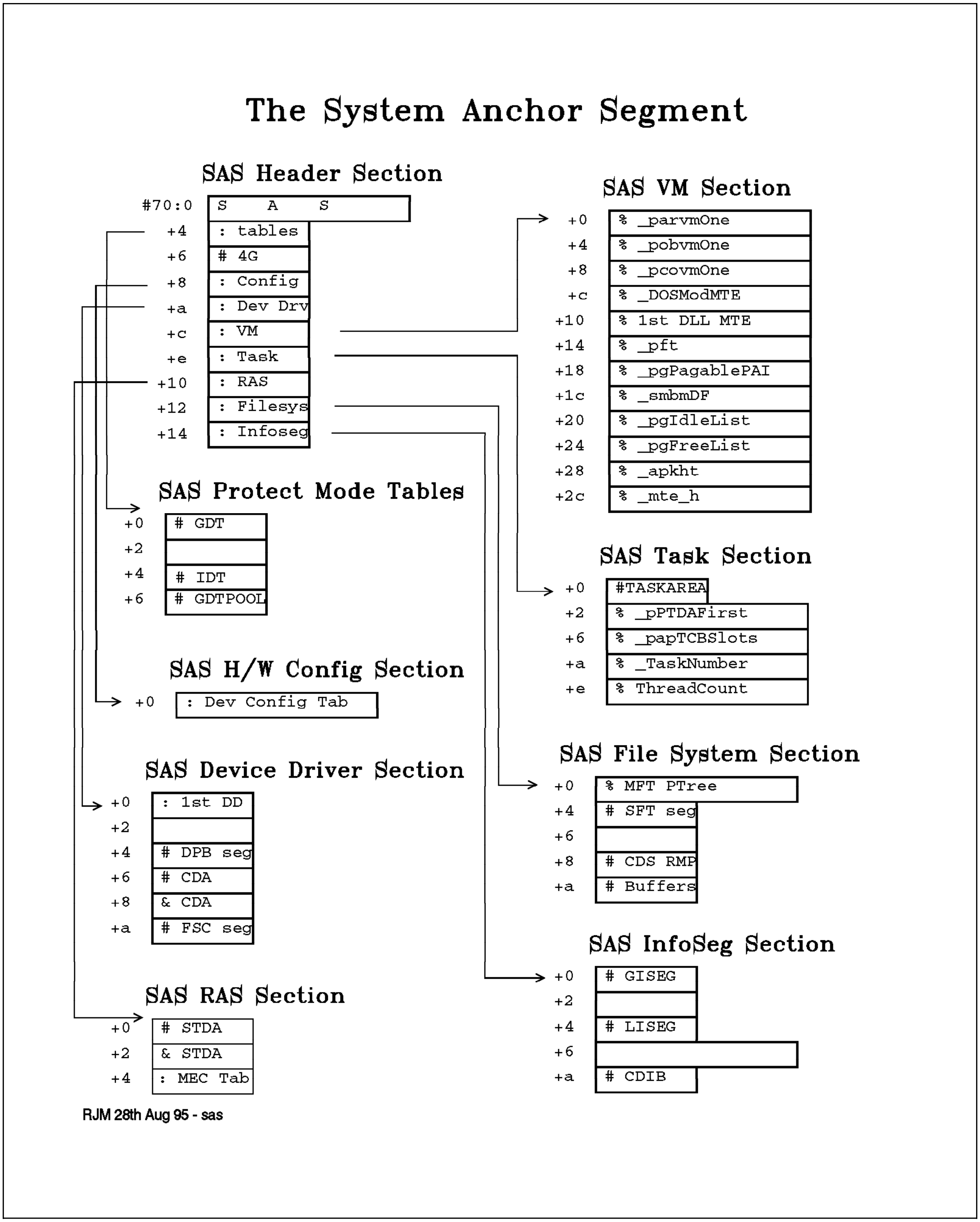


Figure 9. The System Anchor Segment

3.2.2 System Anchor Segment (SAS) for OS/2 Warp V3.0

The SAS is the common anchor for many system control blocks and control block chains.

Pointers

70:0 maps the SAS as a read-only segment.

78:0 maps the SAS as a read/write segment.

Locations

Built statically within the OS2KRNL load module.

VM Owner

os2krnl (0xffaa)

Format

Table 3. SAS Base Section

Field Name	Offset	Length	Type	Description
SAS_signature	+0	4	A	″SAS ″
SAS_tables_data	+4	2	W	offset to tables section
SAS_flat_sel	+6	2	W	FLAT selector for kernel data
SAS_config_data	+8	2	W	offset to configuration section
SAS_dd_data	+a	2	W	offset to device driver section
SAS_vm_data	+c	2	W	offset to Virtual Memory section
SAS_task_data	+e	2	W	offset to Tasking section
SAS_RAS_data	+10	2	W	offset to RAS section
SAS_file_data	+12	2	W	offset to File System section
SAS_info_data	+14	2	W	offset to infoseg section

section.

Table 4. SAS_tables_section Protected Mode Tables

Field Name	Offset	Length	Type	Description
SAS_tbl_GDT	+0	2	W	selector for GDT
SAS_tbl_LDT	+4	2	W	selector for LDT
SAS_tbl_IDT	+6	2	W	selector for IDT
SAS_tbl_GDTPOOL	+8	2	W	selector for GDTPOOL

section.

Table 5. SAS_config_section Configuration Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_config_table	+0	2	W	offset for Device Configuration Table (DevConfigTbl)

Table 6. SAS_dd_section Device Driver Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_dd_bimodal_chain	+0	2	W	offset for the first bimodal device driver′s device header
SAS_dd_real_chain	+2	2	W	offset for the address of the first real mode device driver′s device header
SAS_dd_DPB_segment	+4	2	W	selector for Drive Parameter Block (DPB) segment
SAS_dd_CDA_anchor_p	+6	2	W	selector for ABIOS protected mode Common Data Area
SAS_dd_CDA_anchor_r	+8	2	W	segment for ABIOS real mode Common Data Area
SAS_dd_FSC	+a	2	W	selector for FSC

Table 7 (Page 1 of 2). SAS_vm_section Virtual Memory Management Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_vm_arena	+0	4	D	Flat offset of arena records
SAS_vm_object	+4	4	D	Flat offset of object records
SAS_vm_context	+8	4	D	Flat offset of context records
SAS_vm_krnl_mte	+c	4	D	Flat offset of kernel MTE records
SAS_vm_glbl_mte	+10	4	D	Flat offset of global MTE linked list. Note this field points into the chain to pick up global MTEs only. Use SAS_vm_all_mte to find all the MTEs.
SAS_vm_pft	+14	4	D	Flat offset of page frame table
SAS_vm_prt	+18	4	D	Flat offset of page range table

Table 7 (Page 2 of 2). SAS_vm_section Virtual Memory Management Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_vm_swap	+1c	4	D	Pointer to flat offset of swapper disk frame bit map followed by the size of the bit map in bits WARNING, the bit map offset and size are volatile
SAS_vm_idle_head	+20	4	D	Flat offset of Idle Head
SAS_vm_free_head	+24	4	D	Flat offset of Free Head
SAS_vm_heap_info	+28	4	D	Flat offset of Heap Array
SAS_vm_all_mte	+2c	4	D	Flat offset of all MTEs linked list

Table 8. SAS_task_section Tasking Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_task_PTDA	+0	2	W	selector for current PTDA
SAS_task_ptdaptrs	+2	4	D	FLAT offset for process tree head
SAS_task_threadptrs	+6	4	D	FLAT address for TCB address array
SAS_task_tasknumber	+a	4	D	offset for current TCB number
SAS_task_threadcount	+e	4	D	offset for ThreadCount

Table 9. SAS_RAS_section RAS Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_RAS_STDA_p	+0	2	W	selector for System Trace Data Area (STDA)
SAS_RAS_STDA_r	+2	4	D	segment for System Trace Data Area (STDA)
SAS_RAS_event_mask	+6	4	D	offset for trace event mask

Table 10 (Page 1 of 2). SAS_file_section: File System Section

Field Name	**Offset**	**Length**	**Type**	**Description**
SAS_file_MFT	+0	4	D	handle to MFT PTree
SAS_file_SFT	+4	2	W	selector for System File Table (SFT) segment

Table 10 (Page 2 of 2). SAS_file_section: File System Section

Field Name	Offset	Length	Type	Description
SAS_file_VPB	+ 6	2	W	selector for Volume Parameter Block (VPB) segment
SAS_file_CDS	+ 8	2	W	selector for Current Directory Structure (CDS) segment
SAS_file_buffers	+ a	2	W	selector for buffer segment

Table 11. SAS_info_section Information Segment Section

Field Name	Offset	Length	Type	Description
SAS_info_global	+ 0	2	W	selector for global info seg
SAS_info_local	+ 2	4	D	address of curtask local infoseg
SAS_info_localRM	+ 6	4	D	address of DOS task's infoseg
SAS_info_CDIB	+ a	2	W	selector for Codepage Data

3.2.3 Record Management Package (RMP) for OS/2 Warp V3.0

The RMP is used to manage tables of variable length entities. It appears in a number of situations, particularly those that required ASCII strings, such as file names, to be managed.

Pointers

rp_selector of the RMP handle maps the RMP segment.

Locations

RMP handles are located at the following labels:

CharDevRMPRec	Character Device Drivers
SpoolDevRMPRec	Spooler Device Drivers
NmpRmpHand	Named Pipes
hDiscSegRmpStruc	Discardable Segments
ShareRmpStruc	Named Shared Memory
SysSemRmpHdl	System Semaphores

VM Owner

CharDevRMPRec	**chardevrmp (0xff35)**
SpoolDevRMPRec	**spldevrmp (0xff34)**
NmpRmpHand	**npipenpn (0xff30)**

hDiscSegRmpStruc	**discard (0xff6c)**
ShareRmpStruc	**mshrmp (0xff83)**
SysSemRmpHdl	**syssemrmp (0xff36)**

Format

Table 12. rbheadr RMP Header Structure

Field Name	**Offset**	**Length**	**Type**	**Description**
rb_size	+0	2	W	total size of segment
rb_free_size	+2	2	W	amount of free space
rb_1st_free	+4	2	W	link to first free block in seg
rb_last_free	+6	2	W	start of last free block
rb_hkh	+8	4	D	heap handle
rb_flags	+c	4	D	PG alloc/realloc flags
rb_hobowner	+10	2	W	hobowner
rb_hobmte	+12	2	W	hobmte
rb_first	+14	n	S	start of first record
rb_sz_field	+n+0	2	W	size of ′record size field′
	+n+2	n-2	S	record data

Table 13. rbfree RMP Free Record Structure

Field Name	**Offset**	**Length**	**Type**	**Description**
rf_size	+0	2	W	free block size (high bit set)
rf_prev_free	+2	2	W	link to prev free block in seg
rf_next_free	+4	2	W	link to next free block in seg

Table 14. rparm RMP Handle Structure

Field Name	**Offset**	**Length**	**Type**	**Description**
rp_flags	+0	1	B	flags
	+1	1	B	unused
rp_selector	+2	2	W	GDT selector to use

Table 15. rp_flags Flag Definitions

Name	**Bit Mask**	**Description**
RPF_BUSY	0x01	Segment busy flag
RPF_WAITING	0x02	Somebody waiting flag
RPF_ALLOC	0x04	Segment allocated flag

3.3 Semaphore Control Block Reference

The following control blocks are described in this section:

3.3.1, “FastSafeRamSemStruc”

3.3.2, “FastSafeRamSemStruc PM Version”

3.3.3, “MuxTableEntry” on page 56

3.3.4, “RamSemStruc” on page 56

3.3.5, “KSEM Structures for OS/2 Warp V3.0 ALLSTRICT Kernel” on page 57

3.3.6, “32-Bit Semaphore Structures for OS/2 Warp V3.0 ALLSTRICT Kernel” on page 59

3.3.7, “System Semaphore Structures” on page 66

3.3.8, “PM/GRE Semaphore Structure” on page 68

3.3.1 FastSafeRamSemStruc

Pointers
TCB_SemInfo points to **fs_RAMSem**

Locations
Multiple, in user storage.

VM Owner
Multiple user storage owners.

Format

Field Name	Offset	Length	Type	Description
FastSafeRamSemStruc	-a	e	S	Fast Safe Ram Semaphore
fs_Length	-a	2	W	Length of this structure
fs_ProcID	-8	2	W	Process ID of owner or zero
fs_ThrdID	-6	2	W	Thread ID of owner or zero
fs_Usage	-4	2	W	reference count
fs_Client	-2	2	W	16-bit field for use by owner
fs_RAMSem	+0	4	S	OS/2 RAM Semaphore

3.3.2 FastSafeRamSemStruc PM Version

Pointers
TCB_SemInfo points to **fs_RAMSem**

Locations
Multiple, in user storage.

VM Owner
Multipl user storage owners.

Format

Field Name	Offset	Length	Type	Description
FastSafeRamSemStruc	-e	12	S	PM Fast Safe Ram Semaphore
fs_Length	-e	2	W	Length of this structure
fs_ProcID	-c	2	W	Process ID of owner or zero
fs_ThrdID	-a	2	W	Thread ID of owner or zero
fs_Usage	-8	2	W	reference count
fs_Client	-6	2	W	16-bit field for use by owner
fs_Timeout	-4	4	D	Timeout value
fs_RAMSem	+ 0	4	S	OS/2 RAM Semaphore

3.3.3 MuxTableEntry

Locations

At label **MuxTable** in system storage

VM Owner

os2krnl (0xffaa)

Format

Field Name	Offset	Len	Type	Description
MuxTableEntry	+ 0	9	S	Mux Table Entry
MuxLink	+ 0	2	W	Selector Link to next entry. Used to chain entries for a MuxWait request
MuxThreadID	+ 2	2	W	Thread Slot ID of waiter
MuxType	+ 4	1	B	Semaphore type.
MuxSemID	+ 5	4	D	Mux Semaphore handle.

Table 16. MuxType Flag Definitions

Name	Value	Description
MUXTYPE_CLEAR	0	the mux table entry is clear
MUXTYPE_SYSSEM	1	the ID is a system sem address
MUXTYPE_RAMHANDLE	2	the ID is a ram sem handle:offset
MUXTRYE_RAMPHYS	3	the ID is a ram sem physical address
MUXTYPE_EVENTSEM	4	the ID for a 32-bit event sem

3.3.4 RamSemStruc

Pointers

TCB_SemInfo

Locations

Multiple, in user storage.

VM Owner
Multiple user storage owners.

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
RamSemStruc	+ 0	4	S	Ram Semaphore
RamSemOwner	+ 0	1	B	Ownership flag
RamSemFlag	+ 1	1	B	Ram Semaphore flag bit field
RamSemID	+ 2	2	W	RamSem Block/Run ID low word

Table 17. RamSemFlag Definitions

Name	**Value**	**Description**
RAMSEM_WAITING	0x01	a thread is waiting on the sem
RAMSEM_MUXWAITING	0x02	a thread is muxwaiting on the sem

Notes:

The high-order 4 bit of the **RamSemFlag** are used as an extended owner field (to cater for more than 512 threads).

Only kernel code sets the **RamSemOwner** field to a thread slot number. Ring 3 **RamSems** have **0xff** value for an owned **RamSem**

3.3.5 KSEM Structures for OS/2 Warp V3.0 ALLSTRICT Kernel

For **KSEM** formats for other versions of OS/2 see:

3.3.5.1, "KSEM Structures for OS/2 Warp V3.0 RETAIL Kernel" on page 58

Locations
Multiple, either imbedded in system structures, for example PTDA, MFT, or dynamically allocated from the kernel heaps.

VM Owner
Imbedded KSEMs assume the Owner Id of the imbedding structure.
Stand-alone KSEMs allocated from the kernel heaps use id: **ksem (0xff7e)**

Format

Table 18 (Page 1 of 2). KSEMSHR Shared Kernel Semaphore

Field Name	**Offset**	**Length**	**Type**	**Description**
ks_Signature	+ 0	4	D	
ks_bFlags	+ 4	1	B	
ks_bType	+ 5	1	B	
ks_Owner	+ 6	2	W	
ks_cusPendingWriters	+ 8	2	W	

Table 18 (Page 2 of 2). KSEMSHR Shared Kernel Semaphore

Field Name	Offset	Length	Type	Description
ks_cusNest	+ a	2	W	
ks_cusReaders	+ c	2	W	
ks_cusPendingReaders	+ e	2	W	

Table 19. KSEMMTX MUTEX Kernel Semaphore

Field Name	Offset	Length	Type	Description
ksm_Signature	+ 0	4	D	
ksm_bFlags	+ 4	1	B	
ksm_bType	+ 5	1	B	
ksm_Owner	+ 6	2	W	
ksm_cusPendingWriters	+ 8	2	W	
ksm_cusNest	+ a	2	W	

Table 20. KSEMEVT Event Kernel Semaphore

Field Name	Offset	Length	Type	Description
kse_Signature	+ 0	4	D	
kse_bFlags	+ 4	1	B	
kse_bType	+ 5	1	B	
kse_Owner	+ 6	2	W	
kse_cusPendingWriters	+ 8	2	W	

Table 21. Ksem Flag Definitions

Name	Bit Mask	Description
KSEM_NOINTERRUPT	0x1	
KSEM_WRITER	0x2	
KSEM_DISPLAYID	0x4	
KSEM_NOBLOCK	0x8	

3.3.5.1 KSEM Structures for OS/2 Warp V3.0 RETAIL Kernel

Table 22 (Page 1 of 2). KSEMSHR Shared Kernel Semaphore

Field Name	Offset	Length	Type	Description
ks_bFlags	+ 0	1	B	
ks_bType	+ 1	1	B	

Table 22 (Page 2 of 2). KSEMSHR Shared Kernel Semaphore

Field Name	Offset	Length	Type	Description
ks_Owner	+ 2	2	W	
ks_cusPendingWriters	+ 4	2	W	
ks_cusNest	+ 6	2	W	
ks_cusReaders	+ 8	2	W	
ks_cusPendingReaders	+ a	2	W	

Table 23. KSEMMTX MUTEX Kernel Semaphore

Field Name	Offset	Length	Type	Description
ksm_bFlags	+ 0	1	B	
ksm_bType	+ 1	1	B	
ksm_Owner	+ 2	2	W	
ksm_cusPendingWriters	+ 4	2	W	
ksm_cusNest	+ 6	2	W	

Table 24. KSEMEVT Event Kernel Semaphore

Field Name	Offset	Length	Type	Description
kse_bFlags	+ 0	1	B	
kse_bType	+ 1	1	B	
kse_Owner	+ 2	2	W	
kse_cusPendingWriters	+ 4	2	W	

3.3.6 32-Bit Semaphore Structures for OS/2 Warp V3.0 ALLSTRICT Kernel

For 32-bit Semaphore formats for other versions of OS/2 see:

3.3.6.1, “32-bit Semaphore Structures for OS/2 Warp V3.0 RETAIL Kernel” on page 64

Pointers

TCB_SleepId points to **SEVENT**, **PEVENT**, **SMUTEX**, **PMUTEX**, **SMUX** or **PMUX** when waiting on the semaphore.

PTDA field **pPrSemTbl** points to the private semaphore table, which is indexed by the semaphore handle.

pShSemTbl points to the shared semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structure.

PTDA field **pPrSemTbl** points to the per-process private semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structure.

pShSemStrTbl points to the table of **SEMTBLNODE** entries. Each of these points to a hashed chain of **SEMSTRNODE** structures.

Note: Names are hashed by treating each name as table of null padded ULONGs and successively adding.

Locations
Structures are allocated from the kernel heaps.

VM Owners

SEVENT	**semstruc (0xffc2)**
PEVENT	**semstruc (0xffc2)**
SMUTEX	**semstruc (0xffc2)**
PMUTEX	**semstruc (0xffc2)**
SMUX	**semstruc (0xffc2)**
PMUX	**semstruc (0xffc2)**
OPENQ	**semopenq (0xffbf)**
MUXQ	**semmuxq (0xffbe)**
SEMRECORD	**semrec (0xffc0)**
SEMTBLNODE	**semtable (0xffc3)**
SEMSTRNODE	**semtable (0xffc3)**
Semaphore name	**semstr (0xffc1)**

Format

Table 25. SEVENT Shared Event Semaphore

Field Name	Offset	Length	Type	Description
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenQ	+8	4	D	pointer to the open queue
pszName	+c	4	D	name of semaphore, null if anonymous
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x54564553 ″SEVT″
ptcb	+18	4	D	ptcb of caller

Table 26. PEVENT Private Event Semaphore				
Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenCt	+8	2	W	number of opens
pulCreatAddr	+a	4	D	Address passed in by app during create
ulSig	+e	4	D	0x54564550 ″PEVT″

Table 27. SMUTEX Shared Mutex Semaphore				
Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
pOpenQ	+c	4	D	pointer to the open queue
pszName	+10	4	D	name of semaphore, null if anonymous
pulCreatAddr	+14	4	D	Address passed in by app during create
ulSig	+18	4	D	0x58544D53 ″SMTX″

Table 28. PMUTEX Private Mutex Semaphore				
Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
usOpenCt	+c	2	W	number of opens
pulCreatAddr	+e	4	D	Address passed in by app during create
ulSig	+12	4	D	0x58544D50 ″PMTX″

Table 29. SMUX Shared Mux Wait Semaphore				
Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+0	2	W	attributes
cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
pOpenQ	+a	4	D	pointer to the open queue
pszName	+e	2	W	name of semaphore, null if anonymous
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x58554D53 ″SMUX″

Table 30. PMUX Private Mux Wait Semaphore				
Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+0	2	W	attributes
cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
usOpenCt	+a	2	W	number of opens
pPTDA	+c	4	D	pointer to PTDA of creator
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x58554D50 ″PMUX″

Table 31. OPENQ Open Queue Node Structure				
Field Name	**Offset**	**Length**	**Type**	**Description**
pidOpener	+0	2	W	process id of opening process
usOpenCt	+2	2	W	number of Opens for this process
pNextOpen	+4	4	D	pointer to next node in list
ulSig	+8	4	D	0x514E504F ″OPNQ″

Table 32. MUXQ Mux Queue Node Structure

Field Name	Offset	Length	Type	Description
pMux	+ 0	4	D	pointer to a mux (shared or private)
pNextMux	+ 4	4	D	pointer to next mux waiter in list
ulSig	+ 8	4	D	0x5158554D ″MUXQ″

Table 33. SEMRECORD Semaphore Record Structure for MUX Wait Semaphores

Field Name	Offset	Length	Type	Description
hsemCur	+ 0	4	D	semaphore handle
ulUser	+ 4	4	D	user value

Table 34. SEMSTRNODE Semaphore String Node

Field Name	Offset	Length	Type	Description
hsem	+ 0	4	D	semaphore handle
psz	+ 4	4	D	pointer to the string
pNext	+ 8	4	D	pointer to next string node
ulSig	+ c	4	D	0x444F4E53 ″SNOD″

Table 35. SEMTBLNODE Semaphore String Node Table Entry

Field Name	Offset	Length	Type	Description
ulKey	+ 0	4	D	hash key
pStrNode	+ 4	4	D	pointer to string node

Table 36 (Page 1 of 2). usFlags Field Definitions

Name	Bit Mask	Description
DE_POSTED	0x0040	The event sem APIs set this flag if the event is in the posted state
DM_OWNER_DIED	0x0080	The process died while owning the mutex semaphore
DMW_MTX_MUX	0x0100	The muxwait semaphore APIs set this flag if the mux contains mutex sems

Table 36 (Page 2 of 2). usFlags Field Definitions		
Name	**Bit Mask**	**Description**
DHO_SEM_OPEN	0x0200	dh_OpenEventSem sets this flag to indicate that device drivers have opened the given semaphore
DE_16BIT_MW	0x0400	Part of a 16-bit MuxWait if this flag is set

3.3.6.1 32-bit Semaphore Structures for OS/2 Warp V3.0 RETAIL Kernel

Table 37. SEVENT Shared Event Semaphore

Field Name	Offset	Length	Type	Description
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenQ	+8	4	D	pointer to the open queue
pszName	+c	4	D	name of semaphore, null if anonymous
ptcb	+10	4	D	ptcb of caller

Table 38. PEVENT Private Event Semaphore

Field Name	Offset	Length	Type	Description
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenCt	+8	2	W	number of opens

Table 39. SMUTEX Shared Mutex Semaphore

Field Name	Offset	Length	Type	Description
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
pOpenQ	+c	4	D	pointer to the open queue
pszName	+10	4	D	name of semaphore, null if anonymous

Table 40. PMUTEX Private Mutex Semaphore

Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+ 0	2	W	attributes
pMuxQ	+ 2	4	D	pointer to the mux queue
usRequestCt	+ 6	2	W	number of requests
usSlotNum	+ 8	2	W	slot number of the owning thread
usRequesterCt	+ a	2	W	number of requesters
usOpenCt	+ c	2	W	number of opens

Table 41. SMUX Shared Mux Wait Semaphore

Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+ 0	2	W	attributes
cSemRec	+ 2	2	W	count of semaphore records
pSemRec	+ 4	4	D	array of semaphore record entries
usWaitCt	+ 8	2	W	number of threads waiting on the mux
pOpenQ	+ a	4	D	pointer to the open queue
pszName	+ e	2	W	name of semaphore, null if anonymous

Table 42. PMUX Private Mux Wait Semaphore

Field Name	**Offset**	**Length**	**Type**	**Description**
usFlags	+ 0	2	W	attributes
cSemRec	+ 2	2	W	count of semaphore records
pSemRec	+ 4	4	D	array of semaphore record entries
usWaitCt	+ 8	2	W	number of threads waiting on the mux
usOpenCt	+ a	2	W	number of opens
pPTDA	+ c	4	D	pointer to PTDA of creator

Table 43. OPENQ Open Queue Node Structure				
Field Name	**Offset**	**Length**	**Type**	**Description**
pidOpener	+ 0	2	W	process id of opening process
usOpenCt	+ 2	2	W	number of Opens for this process
pNextOpen	+ 4	4	D	pointer to next node in list

Table 44. MUXQ Mux Queue Node Structure				
Field Name	**Offset**	**Length**	**Type**	**Description**
pMux	+ 0	4	D	pointer to a mux (shared or private)
pNextMux	+ 4	4	D	pointer to next mux waiter in list

Table 45. SEMSTRNODE Semaphore String Node				
Field Name	**Offset**	**Length**	**Type**	**Description**
hsem	+ 0	4	D	semaphore handle
psz	+ 4	4	D	pointer to the string
pNext	+ 8	4	D	pointer to next string node

3.3.7 System Semaphore Structures

Pointers

SysSemRmpHdl contains the selector that points the system semaphore names RMP.

Locations

SysSemDataTable is the location of the global system semaphores table. Each entry is a **SysSemTblStruc** structure.

PTDA field **SysSemPTDATbl** is the location of the per-process semaphore table.

PTDA per-process semaphore contains byte-length entities, which are per-semaphore use counts.

The semaphore handle indexes both the per-process and global semaphore tables.

SysSemHighTable locates the table of **SysSemHighTableS** structures.

VM Owner

syssemrmp (0xff36) for the RMP that contains the semaphore names.

Other global tables are owned by **os2krnl (0xffaa)**.

Format

Table 46. SysSemHandleStruc System Semaphore Handle Structure

Field Name	Offset	Length	Type	Description
SysSemHighWord	+ 0	2	W	0x8000 for sys sems
SysSemPTDAIndex	+ 2	2	W	Index into the PTDA open sem table

Table 47. SysSemTblStruc System Semaphore Table Structure

Field Name	Offset	Length	Type	Description
SysSemOwner	+ 0	2	W	thread owning this semaphore
SysSemFlag	+ 2	1	B	system semaphore flag bit field
SysSemRefCnt	+ 3	1	B	number of references to this sys sem
SysSemProcCnt	+ 4	1	B	number of requests for this owner
SysSemPad	+ 5	1	B	pad byte to round structure up to word

SysSemHighTableS System Semaphore Table Extension Structure.

This is an extension of the SysSemTblStruc that is put into high memory so we don′t impact the low data segment. It is only used in protected mode during process/thread termination.

Field Name	Offset	Length	Type	Description
SysSemPidOwner	+ 0	2	W	pid owner, the thread owner has died

SysSemNameStruc System Semaphore Name table structure, managed by an RMP.

Field Name	Offset	Length	Type	Description
SysSemPtr	+ 0	2	W	

Table 48. SysSemFlag Flag Field Definitions

Name	Bit Mask	Description
SYSSEM_WAITING	0x01	a thread is waiting on the sem
SYSSEM_MUXWAITING	0x02	a thread is muxwaiting on the sem
SYSSEM_OWNER_DIED	0x04	the process/thread owning the sem died
SYSSEM_EXCLUSIVE	0x08	indicates a exclusive system semaphore
SYSSEM_NAME_CLEANUP	0x10	name table entry needs to be removed
SYSSEM_THREAD_OWNER_DIED	0x20	the thread owning the sem died
SYSSEM_EXITLIST_OWNER	0x40	the exitlist thread owns the sem

3.3.8 PM/GRE Semaphore Structure

Locations
pmsemaphores locates the table of PM/GRE semaphores.

VM Owner
PMMERGE.DLL **hmte**

Format

Table 49. GRESEM PM/GRE Semaphore

Field Name	Offset	Length	Type	Description
acIdent	+0	7	A	GRESEM or PMSEM
fcSet	+7	1	B	386 Actual Semaphore
ulProcessThread	+8	4	D	owner process and thread id (PTid)
ulNestedUseCount	+c	4	D	# of times same PTid has accessed sem
ulWaitingCount	+10	4	D	# of PTids waiting on semaphore
ulUseCount	+14	4	D	# of times semaphore has been used
ulEventHandle	+18	4	D	Event Handle Semaphore
ulCallerAddr	+1c	4	D	Semaphore Caller

3.4 Memory Management Control Block Reference

The following control blocks are described in this section:

3.4.2, “Page Frame Structure” on page 80
3.4.4, “VM Alias Record” on page 82
3.4.3, “VM Arena Header” on page 81
3.4.5, “VM Arena Record” on page 83
3.4.6, “VM Arena Type Information Record” on page 85
3.4.7, “VM Context Record” on page 86
3.4.8, “VM Object Record” on page 86
3.4.9, “Virtual Page Structure” on page 88

An overview of the memory management control blocks is as follows:

3.4.1 Memory Management Control Block Diagrams

The following diagrams illustrate the relationships between various memory management control blocks:

3.4.1.1, “Virtual Address Space Regions” on page 70
3.4.1.2, “Virtual Address Space Management” on page 71
3.4.1.3, “Private Arena Private Data” on page 72
3.4.1.4, “Private Arena Shared Data” on page 73
3.4.1.5, “Shared Global Data” on page 74
3.4.1.6, “Shared Arena Instance Data” on page 75
3.4.1.7, “Virtual/Physical Page Management - Backed Storage” on page 76
3.4.1.8, “Virtual/Physical Page Management - Swapped Storage” on page 77
3.4.1.9, “CS Alias of Shared Instance Data” on page 78
3.4.1.10, “Memory Alias in Multiple Processes” on page 79

3.4.1.1 Virtual Address Space Regions

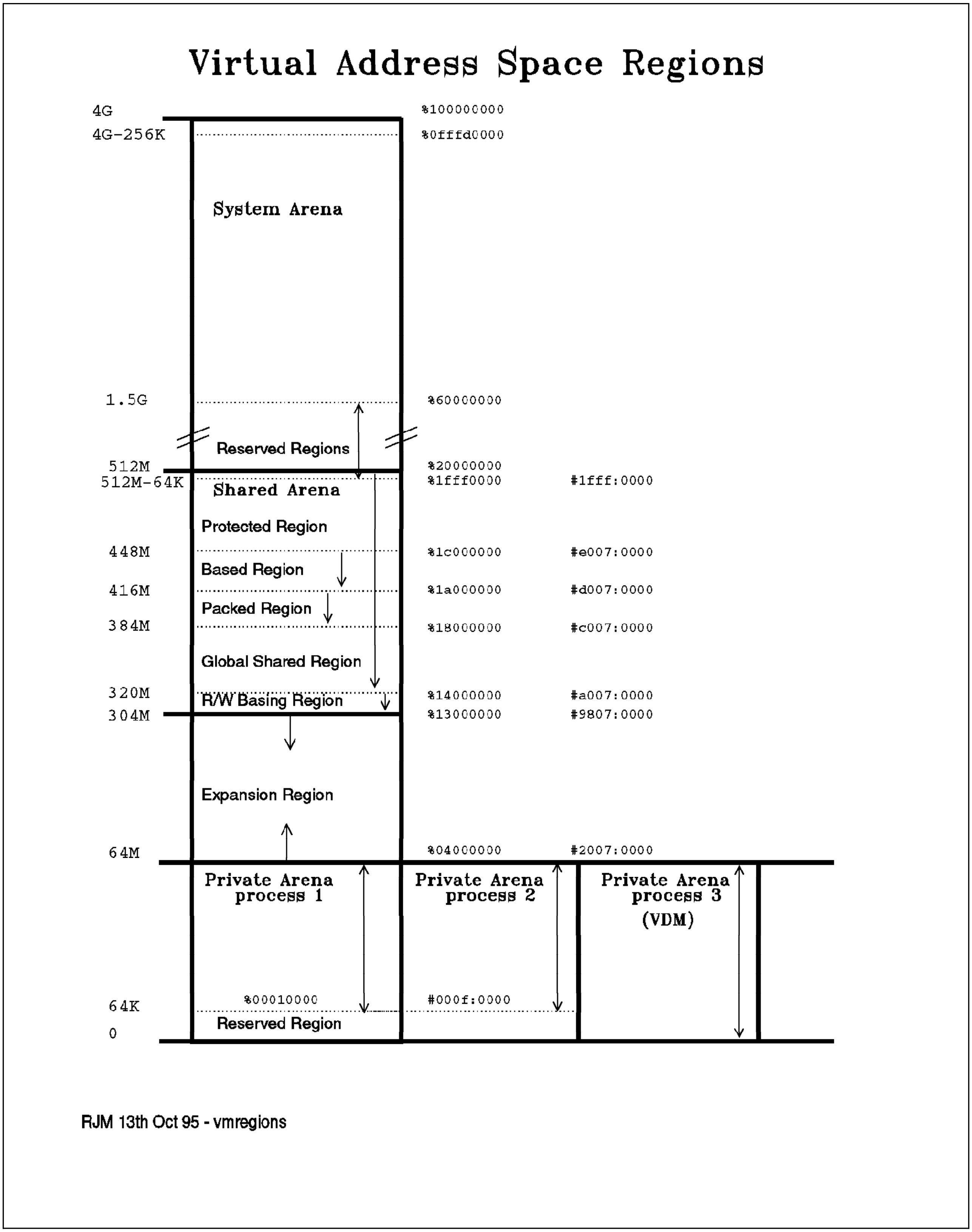


Figure 10. Virtual Address Space Regions

3.4.1.2 Virtual Address Space Management

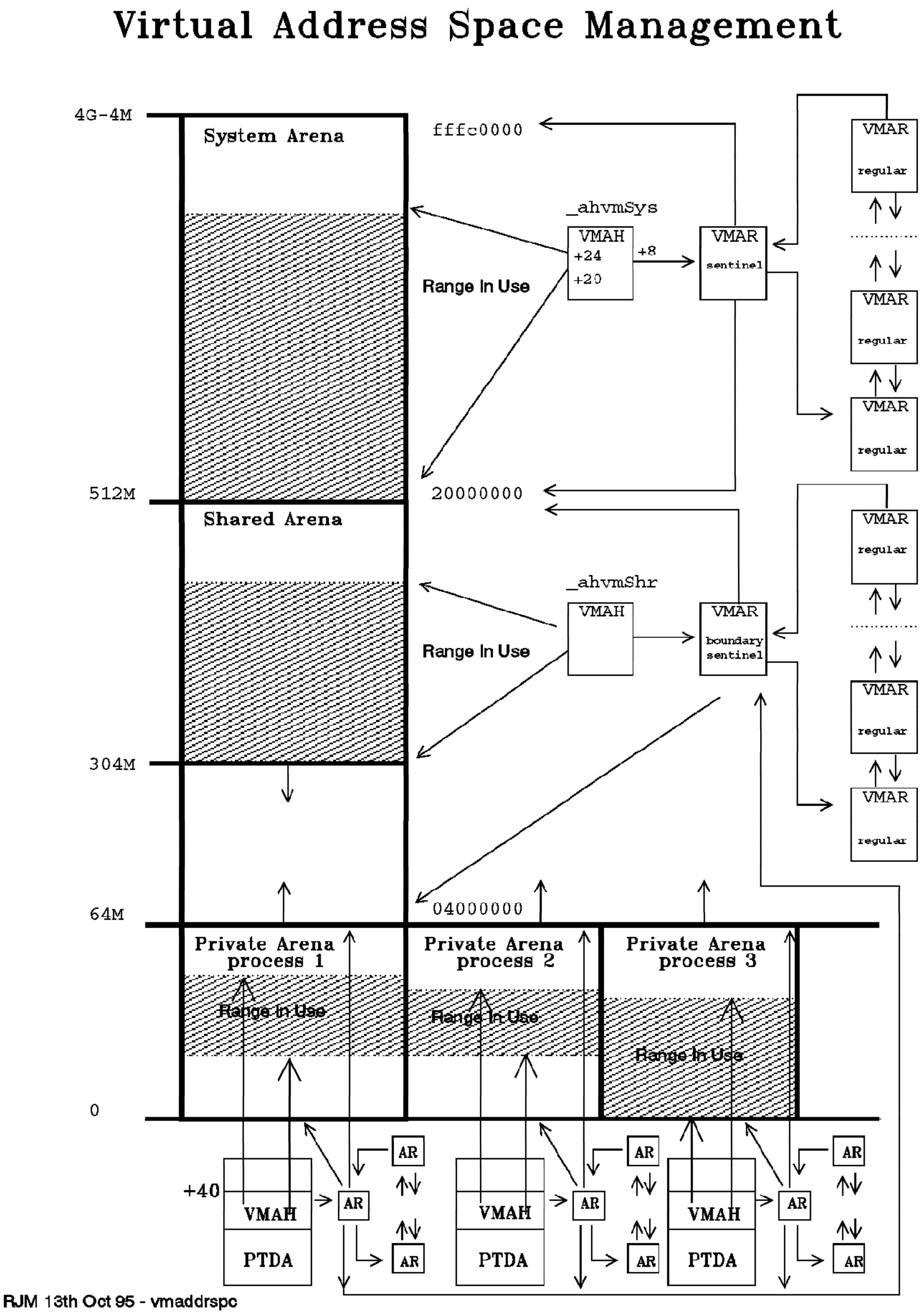


Figure 11. Virtual Address Space Management

3.4.1.3 Private Arena Private Data

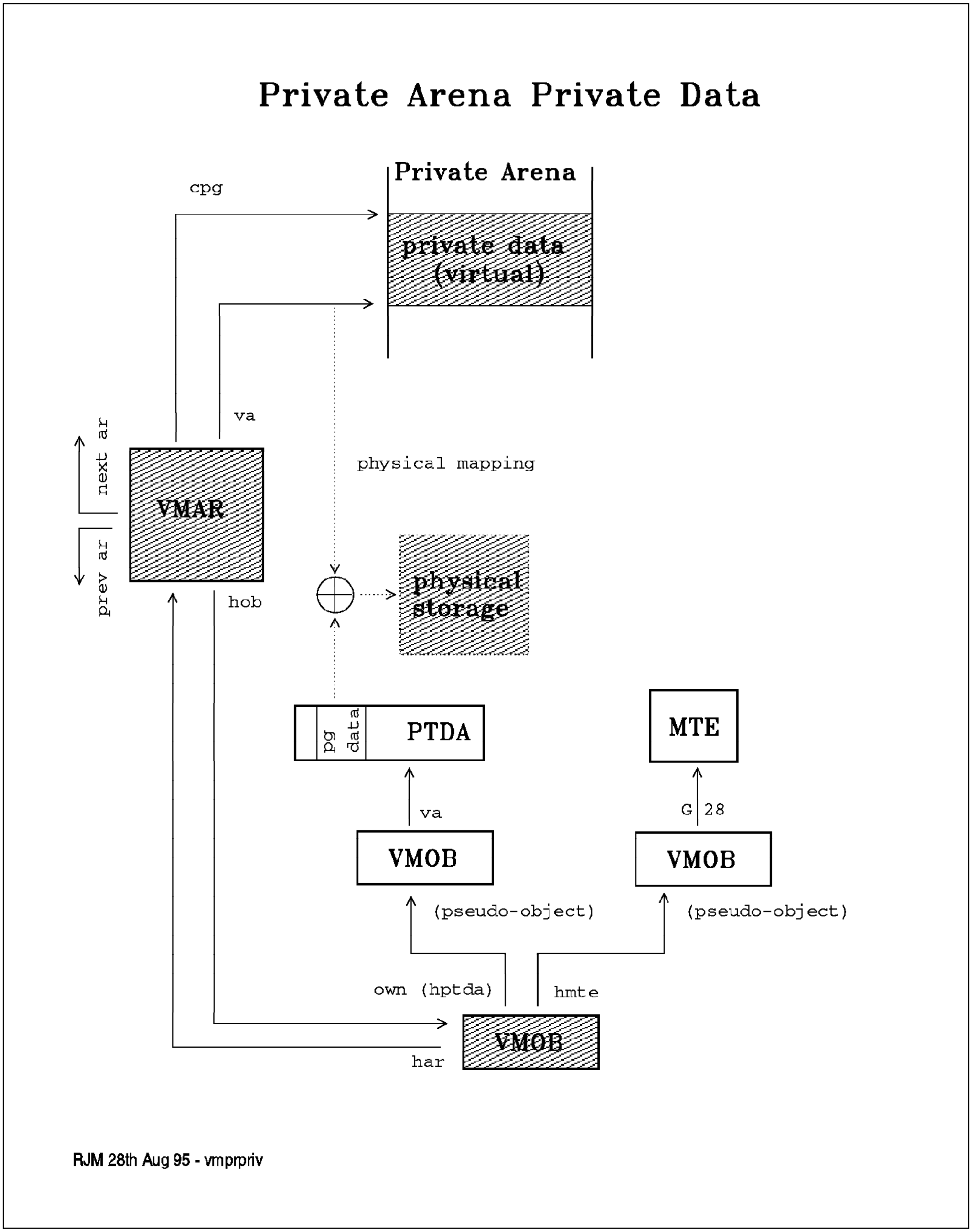


Figure 12. Private Arena Private Data

3.4.1.4 Private Arena Shared Data

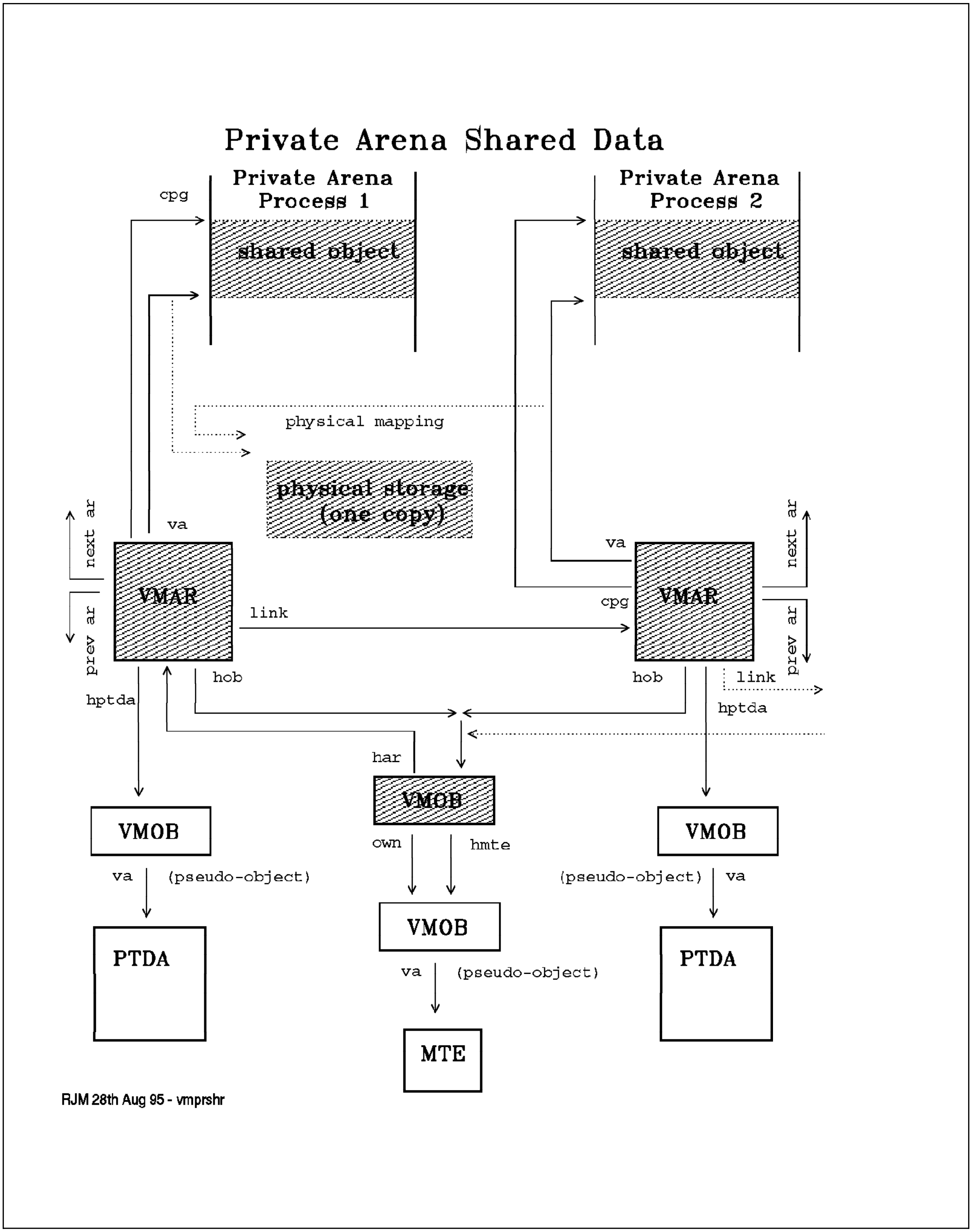


Figure 13. Private Arena Shared Data

3.4.1.5 Shared Global Data

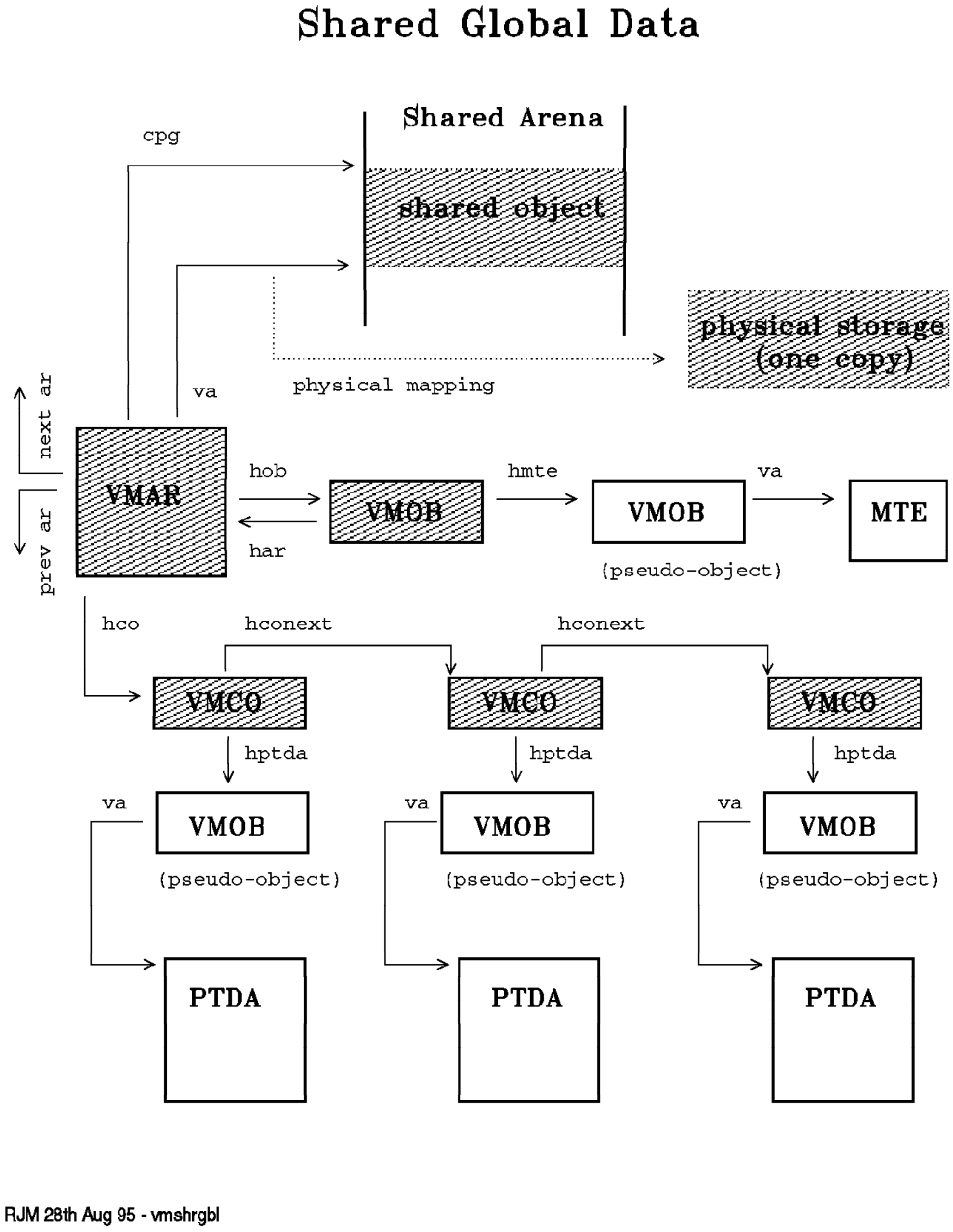


Figure 14. Shared Global Data

3.4.1.6 Shared Arena Instance Data

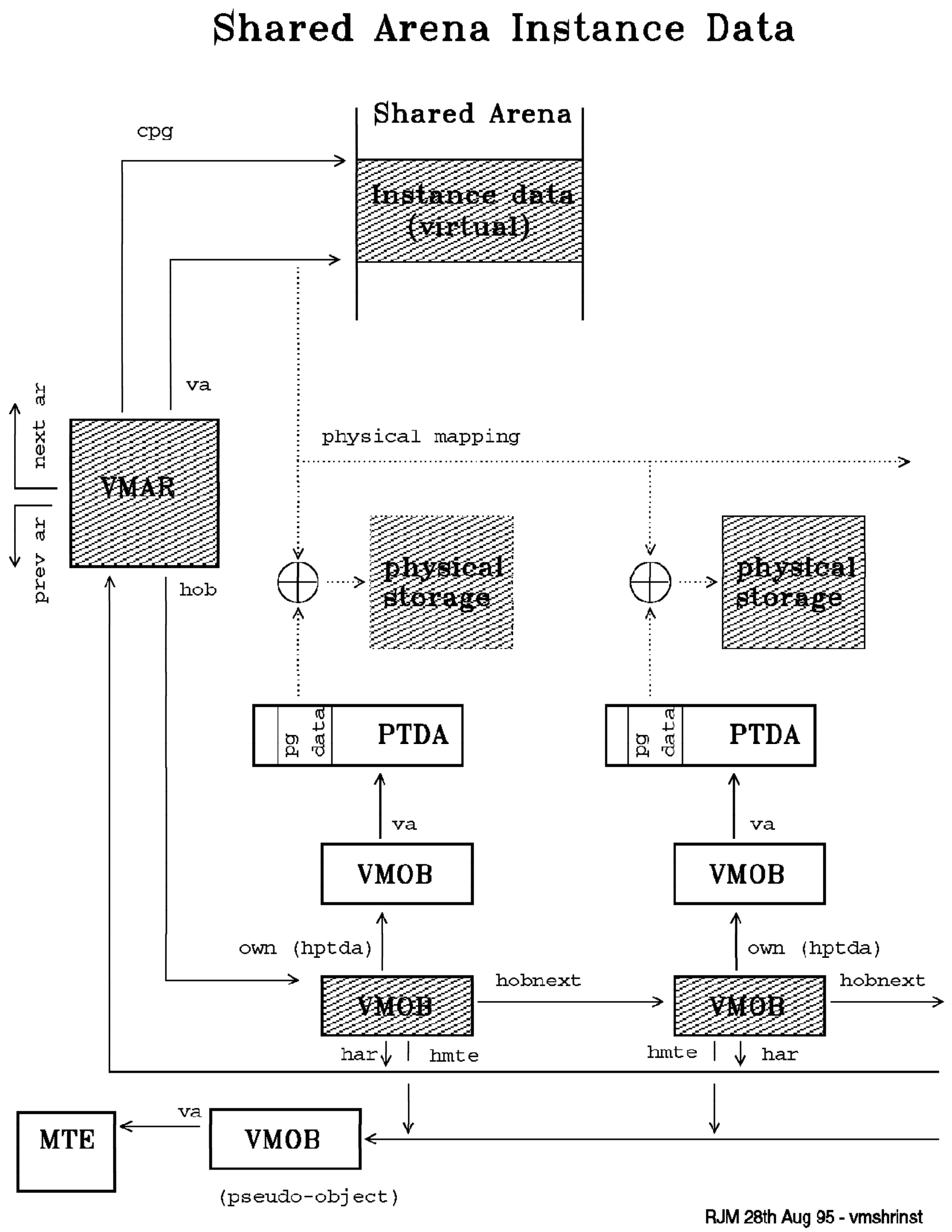


Figure 15. Shared Arena Instance Data

3.4.1.7 Virtual/Physical Page Management - Backed Storage

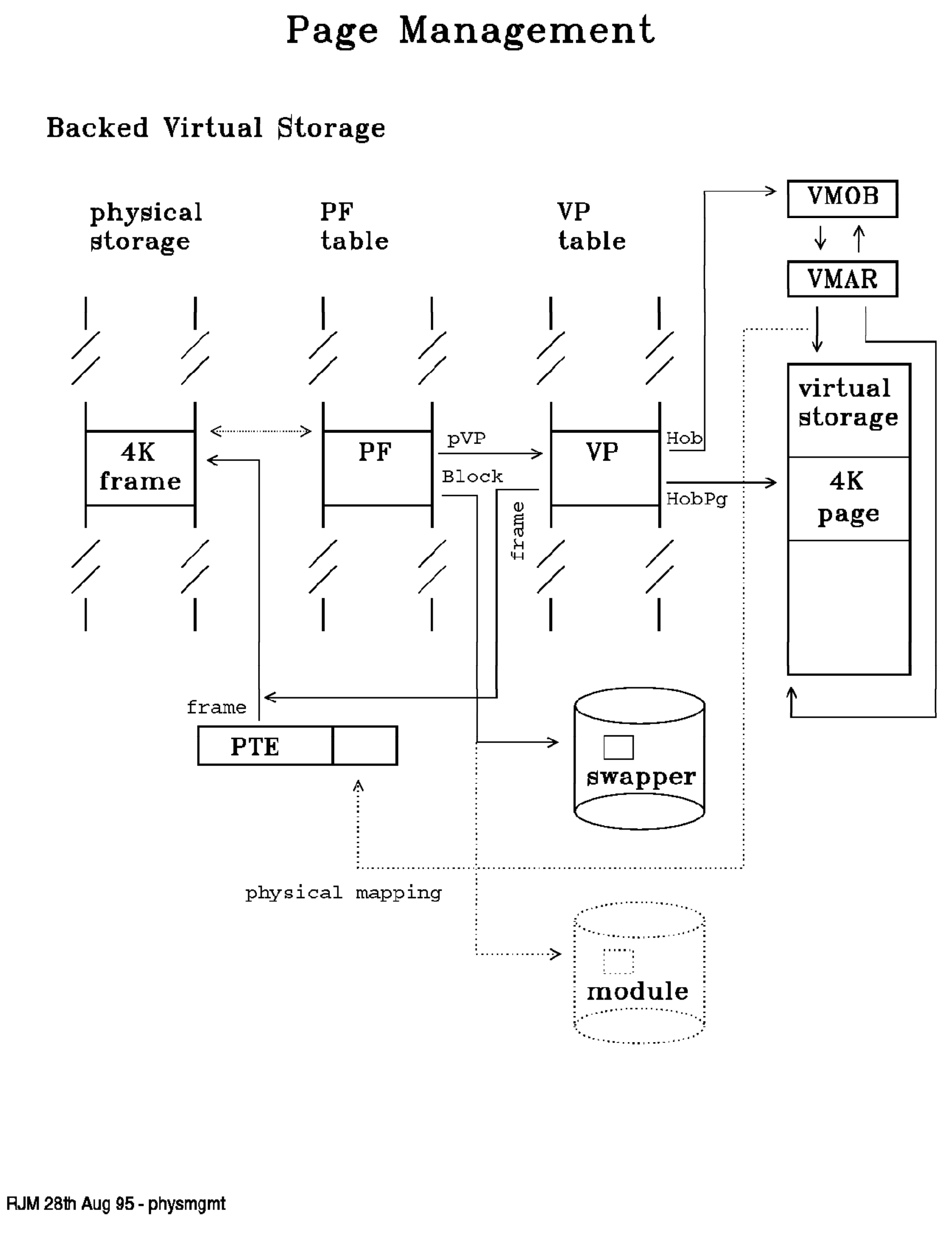


Figure 16. Virtual/Physical Page Management - Backed Storage

3.4.1.8 Virtual/Physical Page Management - Swapped Storage

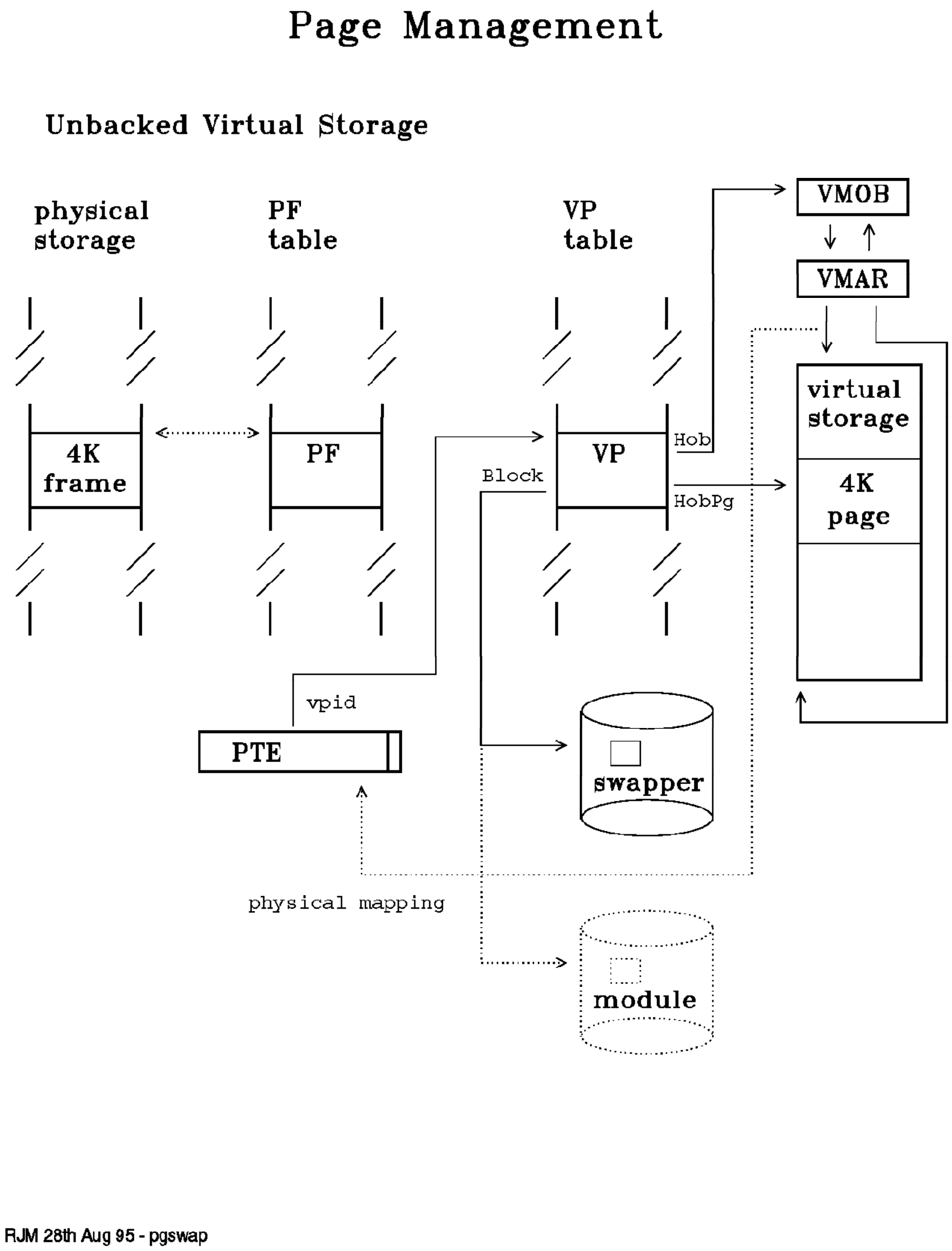


Figure 17. Virtual/Physical Page Management - Swapped Storage

3.4.1.9 CS Alias of Shared Instance Data

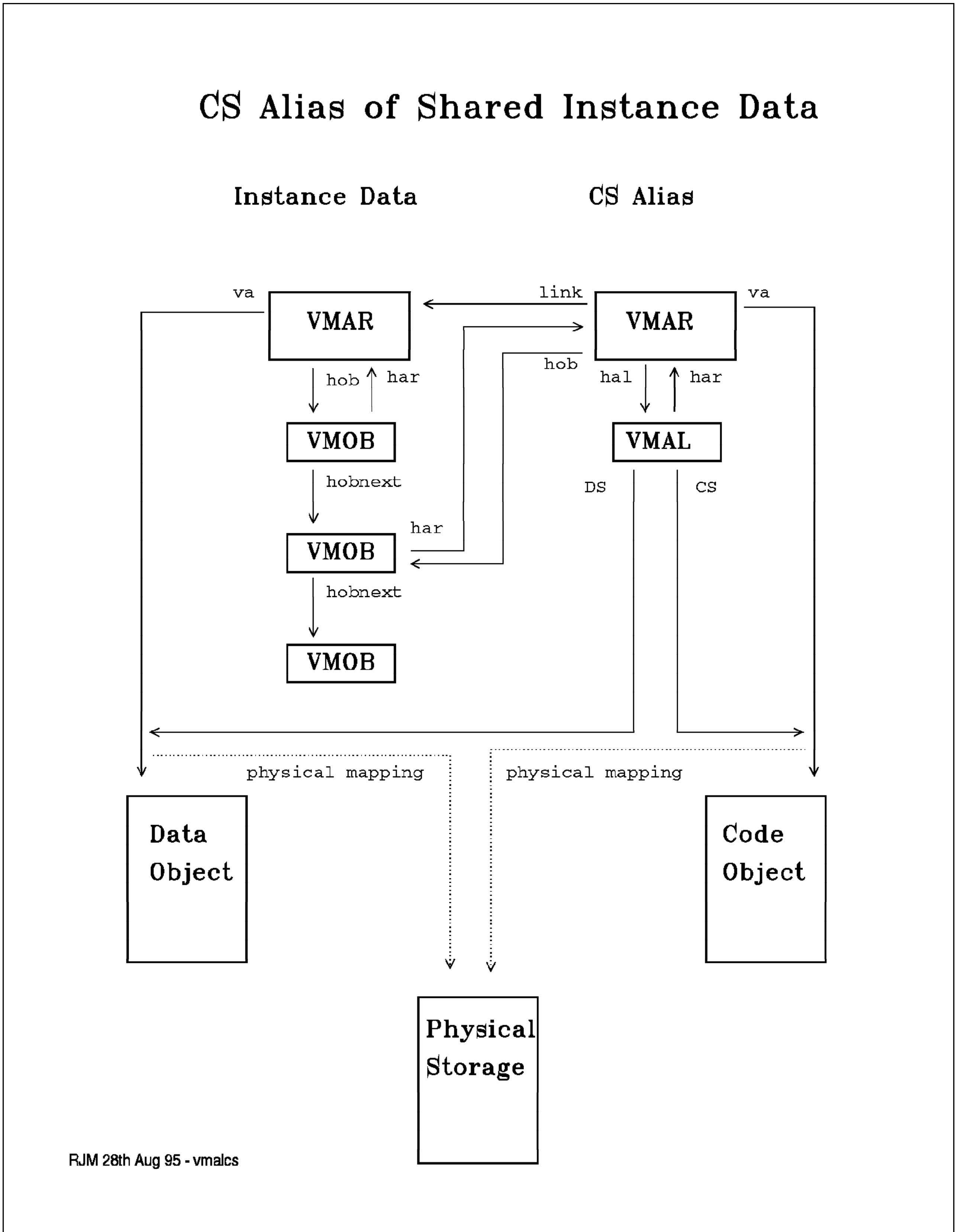


Figure 18. CS Alias of Shared Instance Data

3.4.1.10 Memory Alias in Multiple Processes

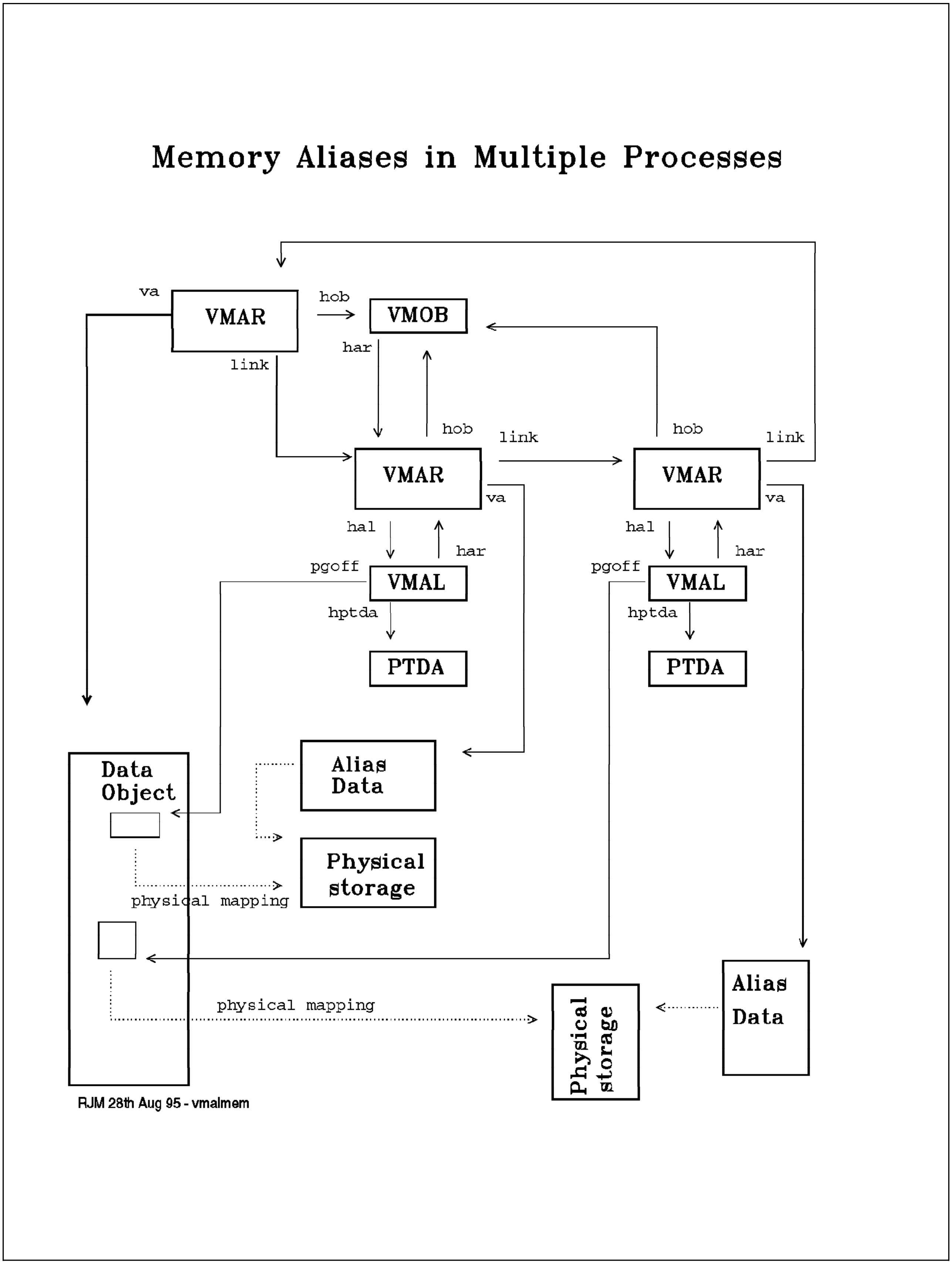


Figure 19. Memory Alias in Multiple Processes

3.4.2 Page Frame Structure

Pointers
_pft points to the table of Page Frame Structures.

Locations
System Arena

VM Owner
pgpf (0xffb4)

Format

Field Name	Offset	Length	Type	Description
apf_s	+0	c	S	active pf
pf_pvp	+0	4	D	vp cross link
pf_refcount	+4	2	W	count of ptes marked present
pf_flags	+6	0.4	B	flags
	+6.4	0.4		pad
pf_llock	+7	1	B	count of long term locks
pf_slock	+8	1	B	count of short term locks
	+9	3	B	pad
ipf_s	+0	c	S	idle page frame
	+0	4	D	pad
pf_flink1	+4	1	D	forward link part 1
pf_flags	+6	0.4		flags
vp_blink	+6.4	2.4		backward link
	+8	1.4	D	pad
pf_flink2	+8.4	2.4		forward link part 2
fpf_s	+0	c	S	free page frame
	+0	4	D	pad
pf_flink1	+4	1	D	forward link part 1
pf_flags	+6	0.4		flags
vp_blink	+6.4	2.4		backward link
	+8	1.4	D	pad
pf_flink2	+8.4	2.4		forward link part 2

Table 50 (Page 1 of 2). pf_flag Flag Definitions

Name	Bit Mask	Description
PF_FAST	0x1	frame is fast memory
PF_BUSY	0x2	frame is busy

Table 50 (Page 2 of 2). pf_flag Flag Definitions		
Name	**Bit Mask**	**Description**
PF_FREE	0x4	frame is free
PF_RES	0x8	reserved

3.4.3 VM Arena Header

Locations

_ahvmSys locates the System Arena VMAH.

_ahvmShr locates the Shared Arena VMAH.

PTDA field **ptda_ah** locates each Private Arena VMAH.

VM Owner

For shared and system arenas: **os2krnl (0xffaa)**

For private arenas: **ptda (0xffcb)**

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
ah_pahNext	+0	4	D	Link to next arena
ah_pahPrev	+4	4	D	Link to previous arena
ah_parSen	+8	4	D	Handle of arena sentinel
ah_parFree	+c	4	D	Hint of 1st free block in arena
ah_papbm	+10	4	D	Pointer to bitmap directory
ah_paharHash	+14	4	D	Hash table pointer
ah_pat	+18	4	D	Pointer to per-type info
ah_fl	+1c	4	D	Flags
ah_laddrMin	+20	4	D	Minimum address currently mapped
ah_laddrMax	+24	4	D	Max address currently mapped
ah_car	+28	4	D	Count of arena entries
ah_carBitmap	+2c	4	D	Max entry count to need bitmap
ah_lbmNumbMax	+30	4	D	Max bitmap number
ah_lbmeNumbMax	+34	4	D	Max bitmap entry number
ah_lHashNumbMax	+38	4	D	Max hash table index
ah_hob	+3c	2	W	Arena header pseudo-handle
ah_filler	+3e	2	W	Make structure 4-byte multiple

Table 51. ah_fl Flag Definitions

Name	**Bit Mask**	**Description**
VMAH_BITMAP_BYPASS	0x00000001	Worth bypassing bitmap
VMAH_NO_HASH_WRAP	0x00000002	No hash table wraparound yet
VMAH_GROW_DOWN	0x00000004	Arena grows down

3.4.4 VM Alias Record

Pointers

_palVMAliases points to the VMAL table.

VM Owner

vmal (0xffe2)

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
vmal	+ 0	8	S	VM alias record
al_har	+ 0	2	W	handle to alias′ arena record
al_hobptda	+ 2	2	W	context the alias is created from
al_pgoff	+ 4	2.4	D	page offset of the alias from start of object
al_f	+ 4	1.4		flags indicating type of alias
vmsal	+ 0	8	S	SEL alias record
sal_har	+ 0	2	W	handle to alias′ arena record
sal_selcode	+ 2	2	W	code selector if cs alias
al_hobptda	+ 2	2	W	context the alias is created from if MEMMAP alias
sal_cref	+ 4	1.2	D	reference count
sal_f	+ 4	0.6		flags
sal_seldata	+ 6	2	W	data selector if cs alias (unused for MEMMAP)

Table 52 (Page 1 of 2). al_f Flag Definitions

Name	Bit Mask	Description
AL_ISBUSY	0x1	Set if record is busy
AL_CSALIAS	0x2	Set if cs alias record
AL_MEMMAP	0x4	Set if MemMapAlias record
AL_DBGALIAS	0x8	Set if debug alias
AL_CSDSVALID	0x10	Set if ds selector valid
AL_DEVHLP	0x20	Set if Devhlp alias
AL_PRIV	0x40	Set if privatized alias

Table 52 (Page 2 of 2). al_f Flag Definitions		
AL_VDM	0x80	Set if VDM alias
AL_NOALIAS	0x100	Set if UVIRT mapping in VDMs

Table 53. sal_f Flag Definitions		
Name	Bit Mask	Description
SAL_CSALIAS	AL_CSALIAS	
SAL_MEMMAPALIAS	AL_MEMMAP	
SAL_CSDSVALID	0x10	should not coincide with other alias types
SAL_ALIASREFSHIFT	0x6	Low six bits reserved for flags
SAL_ALIASREFMASK	0x0ffc0	reference count bits mask

3.4.5 VM Arena Record

Pointers

_parvmOne points to the VMAR table.

VM Owner

vmar (0xffe3)

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
vmar_reg	+0	16	S	Regular Arena Record
ar_xf	+0	1.4	D	Extra flags
ar_cpg	+1.4	2.4		Size in pages
ar_ipg	+4	2.4	D	Virtual page no.
ar_f	+6.4	1.4		Flags
ar_harnext	+8	2	W	Handle of next Arena Record
ar_harprev	+a	2	W	Handle of previous Arena Record
ar_harlink	+c	2	W	Handle of associated Arena Record
ar_harhash	+e	2	W	Hash table link
ar_hob	+10	2	W	Handle of Object Record
ar_hco	+12	2	W	Context record handle (shar+shr data)
ar_hobptda	+12	2	W	PTDA handle or NULL (prvar or shar + instance data)
ar_sel	+12	2	W	Selector (sysarena only)
ar_hal	+14	2	W	Alias record handle, * =0 means not an alias
vmar_sen	+0	16	S	Sentinel Arena Record
ar_xf	+0	1.4	D	Extra flags

Field Name	Offset	Length	Type	Description
ar_cpg	+1.4	2.4		Size in pages
ar_ipg	+4	2.4	D	Virtual page no.
ar_f	+6.4	1.4		Flags
ar_harnext	+8	2	W	Handle of next Arena Record
ar_harprev	+a	2	W	Handle of previous Arena Record
ar_harlink	+c	2	W	Handle of associated Arena Record
ar_harhash	+e	2	W	Hash table link
ar_ipgmax	+10	4	D	Maximum large no. in the arena
ar_unused	+14	2	W	reserved

Table 54. ar_f Flag Definitions

Name	Bit Mask	Description
AR_INUSE	0x001	Record not on free list
AR_TAG	0x006	Record type mask
AR_TAGREG	0x000	Regular record
AR_TAGSEN	0x002	Sentinel
AR_TAGBSEN	0x006	Boundary sentinel
AR_SELMAP	0x008	Memory mapped by selector
AR_SELBASEALL	0x00c	Base selector map all
AR_SELMASK	0x00c	Selector map mask
AR_RELOAD	0x010	Pre-reserved for huge item or
AR_WRITE	0x020	Write permission
AR_USER	0x040	User pages
AR_EXEC	0x080	Executable Pages
AR_READ	0x100	Read permission
AR_HCO	0x200	Record linked to Context List
AR_GUARD	0x400	Guard pages
AR_SGS	0x800	Registered under Screen Group Switch

Table 55. ar_xf Flag Definitions

Name	Bit Mask	Description
AR_HCOH	0x001	context record handle > 64K

3.4.6 VM Arena Type Information Record

Pointers

VMAH field **ah_pat** points to the associated VMAT.

Locations

_atvm locates the table of VMATs.

VM Owner

os2krnl (0xffaa)

Format

Field Name	Offset	Length	Type	Description
at_laddrInitMin	+0	4	D	Initial minimum
at_laddrInitMax	+4	4	D	Initial maximum
at_laddrAbsMin	+8	4	D	Abs minimum boundary
at_laddrAbsMax	+c	4	D	Abs minimum boundary
at_cbInitBetween	+10	4	D	Spacer between arenas
at_lHashNumbMask	+14	4	D	Hash number mask
at_lHashNumbShift	+18	4	D	Hash number shift
at_lHashNumbAbsMax	+1c	4	D	Max hash table index
at_lHashMinSize	+20	4	D	Min hash table size
at_lbmNumbMask	+24	4	D	Bitmap number mask
at_lbmNumbShift	+28	4	D	Bitmap number shift
at_lbmNumbAbsMax	+2c	4	D	Abs Max bitmap #
at_lbdMinSize	+30	4	D	Min bitmap dir size
at_lbmeNumbMask	+34	4	D	Bitmap entry # mask
at_lbmeNumbShift	+38	4	D	Bitmap entry # shift
at_lbmeNumbAbsMax	+3c	4	D	Abs Max bitmap entry
at_lbmeBitNumbMask	+40	4	D	Bit number mask
at_lbmeBitNumbShift	+44	4	D	Bit number shift
at_flInit	+48	4	D	Initial flags
at_lGran	+4c	4	D	Granularity
at_laddrMinNoWrap	+50	4	D	Min no-hash wrap laddr
at_laddrMaxNoWrap	+54	4	D	Max no-hash wrap laddr
at_harParent	+58	2	W	Parent arena

Table 56. at_flInit Flag Definitions

Name	Bit Mask	Description
VMAT_PRIV_TILED	0	
VMAT_PRIV_VDM	1	
VMAT_SHR_TILED	2	
VMAT_SYS	3	
VMAT_MAX	VMAT_SYS	

3.4.7 VM Context Record

Pointers
_pcovmOne points to the table of VMCOs.

Locations
System Arena.

VM Owner
vmco (0xffe5)

Format

Field Name	Offset	Length	Type	Description
co_hconext	+ 0	2	W	Index of next Context Record
co_hobptda	+ 2	2	W	PTDA handle
co_fb	+ 4	1	B	Context record flags

Table 57. co_fb Flag Definitions

Name	Bit Mask	Description
CO_CREATOR	0x01	originating context
CO_PRIV	0x80	Privatized context
CO_HCOH	0x20	Next context record handle > 64K
CO_WRITE	0x02	Write permission
CO_USER	0x04	User storage
CO_EXEC	0x08	Executable
CO_READ	0x10	Read permission
CO_GUARD	0x40	Guard page

3.4.8 VM Object Record

Pointers
_pobvmOne points to the table of VMOBs.

Locations
System Arena.

VM Owner
vmob (0xfff1)

Format

Field Name	Offset	Length	Type	Description
ob_har	+ 0	2	W	Arena Record handle
ob_hobnext	+ 2	2	W	Associated Object Record handle
ob_va	+ 0	4	D	Pseudo-object′s virtual address

Field Name	**Offset**	**Length**	**Type**	**Description**
ob_fs	+ 4	2	W	Flags
ob_hobowner	+ 6	2	W	Owner ID
ob_hobmte	+ 8	2	W	MTE handle
ob_wsemowner	+ a	2	W	ID of thread owning semaphore
ob_bsemcnt	+ c	1	B	Counter and waiting flag
ob_cllock	+ d	1	B	Count of all long-term locks
ob_cslock	+ e	1	B	Count of all short-term locks
ob_xflags	+ f	1	B	Extra flags

Note:

A complete list of system owner IDs may be found under VM System Object Owner IDs in the Reference Tables section of the System Reference.

Table 58. ob_fs Flag Definitions

Name	**Bit Mask**	**Description**
OB_PSEUDO	0x8000	Pseudo-object
OB_API	0x4000	API allocated object
OB_LOCKWAIT	0x2000	Some thread to wake in VMUnlock
OB_LALIAS	0x1000	Object has aliases
OB_SHARED	0x0800	Object′s contents are shared
OB_UVIRT	0x0400	UVirt object
OB_ZEROINIT	0x0200	Object is zero-initialized
OB_RESIDENT	0x0100	Initial allocation was resident
OB_LOWMEM	0x0040	Object is in low memory
OB_GUARD	0x0080	Page attribute/permission flags
OB_EXEC	0x0020	Executable
OB_READ	0x0010	Read permission
OB_USER	0x0008	User Storage
OB_WRITE	0x0004	Write permission
OB_HUGE	0x0002	Object is huge
OB_SHRINKABLE	0x0001	Object is Shrinkable
OB_DHSETMEM	0x0001	DevHlp_VMSetMems are allowed

Table 59. ob_xflags Flag Definitions

Name	**Bit Mask**	**Description**
VMOB_SLOCK_WAIT	0x01	Waiting on short term locks to clear
VMOB_LLOCK_WAIT	0x02	Waiting on long term locks to clear
VMOB_DISC_SEG	0x04	Object is part of a discardable seg
VMOB_HIGHMEM	0x08	Object was allocated via dh_vmalloc

3.4.9 Virtual Page Structure

Pointers

pf_pvp points to the head of the VP array.

Locations

System Arena.

VM Owner

pgvp (0xffb3)

Format

Field Name	Offset	Length	Type	Description
avp_s	+ 0	c	S	active vp
vp_frame	+ 0	2.4	D	frame, swp or ldr block #
vp_flags	+2.4	1.4		flags
vp_obpg	+ 4	2	W	object relative page number
vp_hob	+ 6	2	W	handle to object record
vp_refcount	+ 8	2	W	virtual page reference count
vp_semowner	+ a	2	W	Slot number of semaphore owner
fvp_s	+ 0	a	D	Free vp
vp_flink	+ 0	4	S	forward link
vp_blink	+ 4	4	D	backward link
	+ 8	2	W	pad

Table 60. vp_flag Flag Definitions

Name	Bit Mask	Description
VP_BUSY	0x001	page semaphore taken
VP_WANTED	0x002	page semaphore requested
VP_CACHE	0x004	search page cache for pf
VP_PFIDLE	0x008	cross linked to idle pf
VP_PF	0x010	cross linked to pf
VP_DF	0x020	has swap file disk frame
VP_DIRTY	0x040	contents written to - from pte
VP_SHDIRTY	0x080	shadow dirty bit (for VDMs)
VP_SOW	0x100	change to swappable on write
VP_PRIVATIZED	0x200	vp privatized
VP_RESIDENT	0x400	cannot be moved - value from pte
VP_DISCARDABLE	0x800	1 = discardable, 0 = swappable

3.5 Scheduler Thread and Process Control Block Reference

The following control blocks are described in this section:

- 3.5.2, “Thread Control Block OS/2 Warp V3.0” on page 98
- 3.5.3, “Thread Swappable Data for OS/2 Warp V3.0 ALLSTRICT Kernel” on page 121
- 3.5.4, “Per-Task Data Area for OS/2 Warp V3.0 ALLSTRICT Kernel” on page 125
- 3.5.5, “Local Information Segement” on page 147
- 3.5.6, “Global Information Segement” on page 148
- 3.5.7, “Process Information Block” on page 150
- 3.5.8, “Thread Information Block” on page 151
- 3.5.9, “System Stack Frames Client Register Information” on page 152
- 3.5.10, “Exit List Entry Data Structure” on page 161
- 3.5.11, “Exception Handler Structures” on page 161

An overview of the scheduler control blocks is as follows:

3.5.1 Scheduler and Task Management Control Block Diagrams

The following diagrams illustrate the relationships between various scheduler and task management control blocks:

- 3.5.1.1, “Process Management” on page 90
- 3.5.1.2, “Thread Management” on page 91
- 3.5.1.3, “Scheduler Finite State Machine” on page 92
- 3.5.1.4, “Thread Tree for a Process” on page 93
- 3.5.1.5, “Process Trees, Subtrees and Zombies” on page 94
- 3.5.1.6, “Orphaned and Adopted Processes” on page 95
- 3.5.1.7, “OS/2 Exception Management - Overview” on page 96
- 3.5.1.8, “Exception Handler Stack Frames” on page 97

3.5.1.1 Process Management

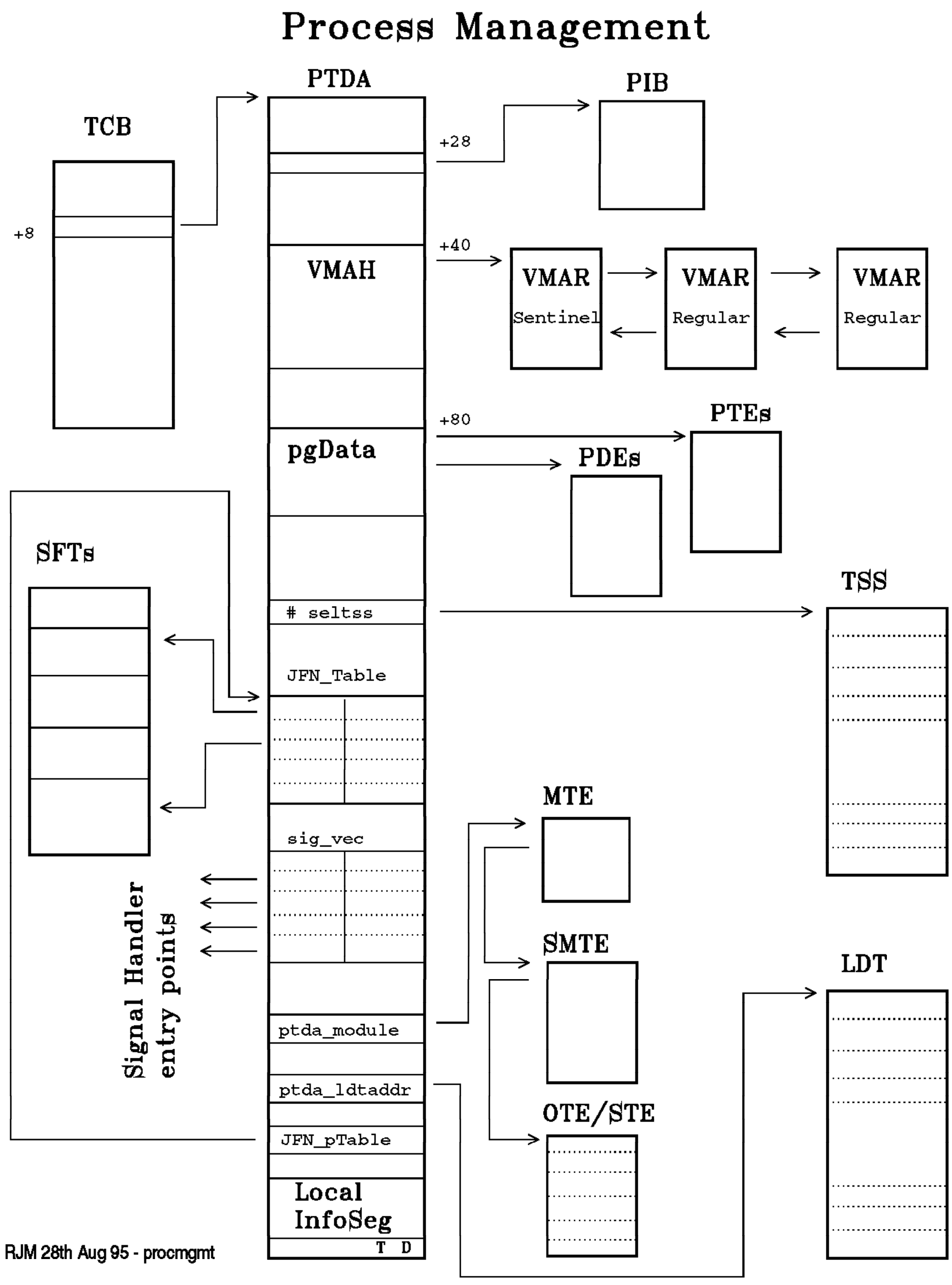


Figure 20. Process Management

3.5.1.2 Thread Management

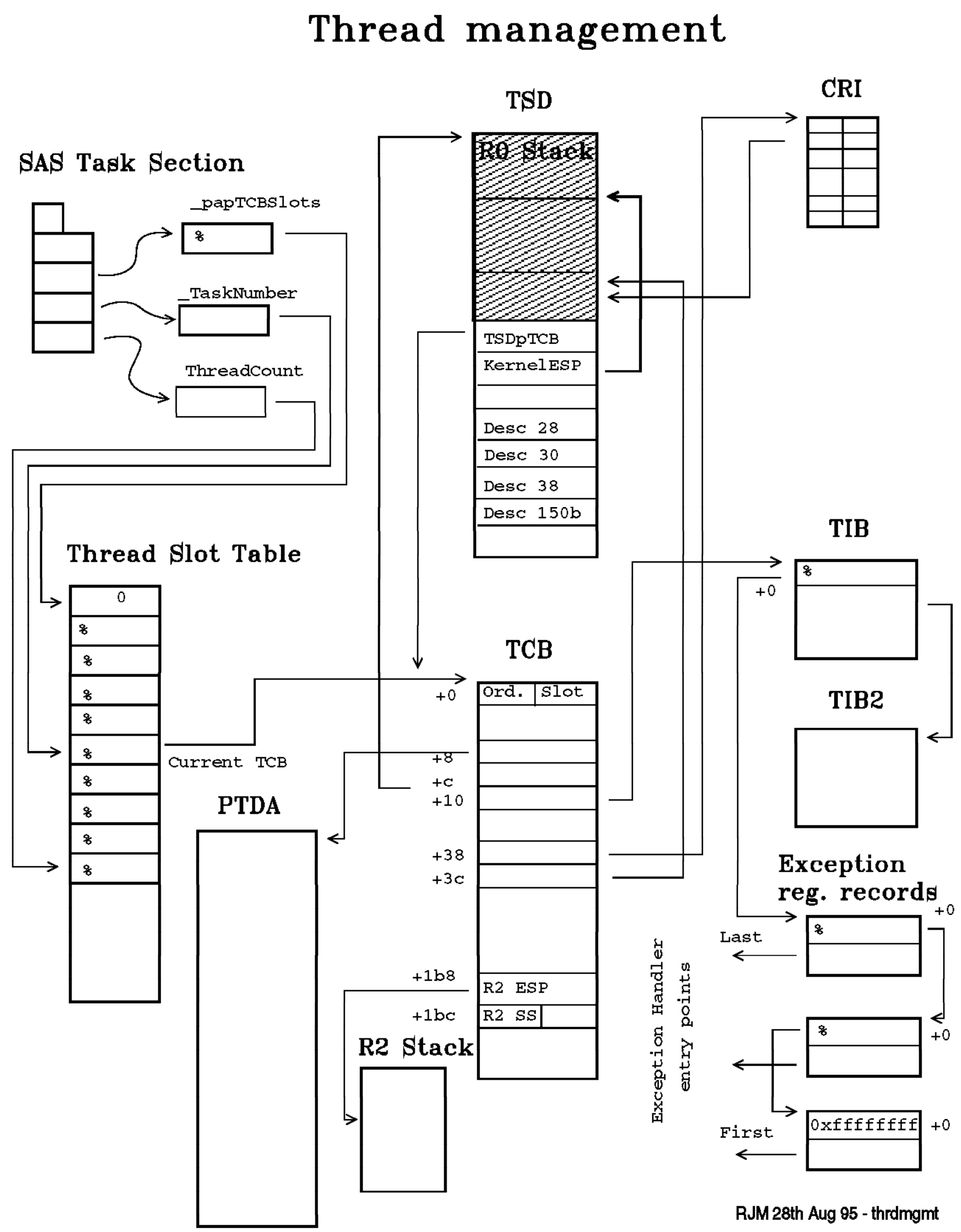


Figure 21. Thread Management

3.5.1.3 Scheduler Finite State Machine

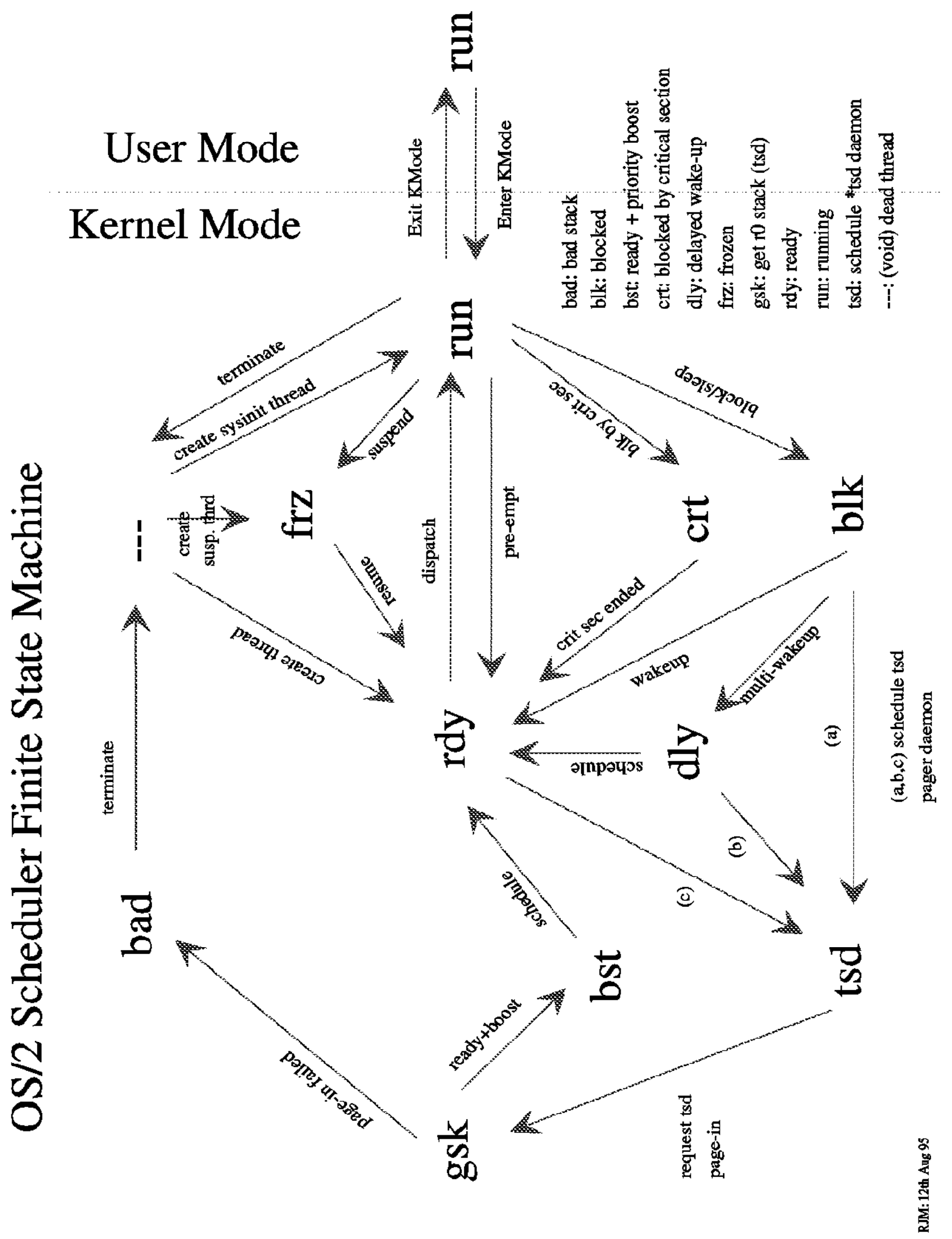


Figure 22. Scheduler Finite State Machine

3.5.1.4 Thread Tree for a Process

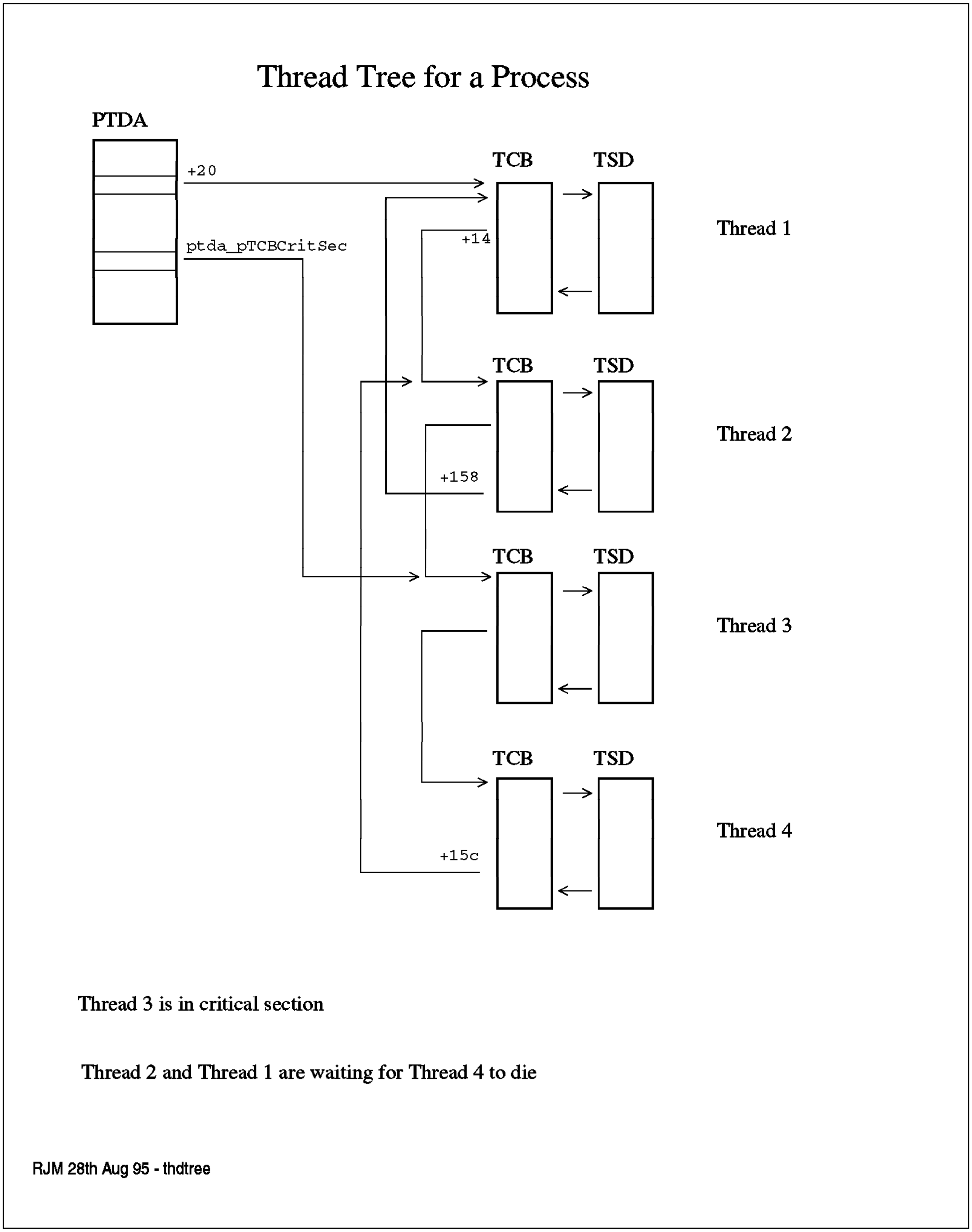
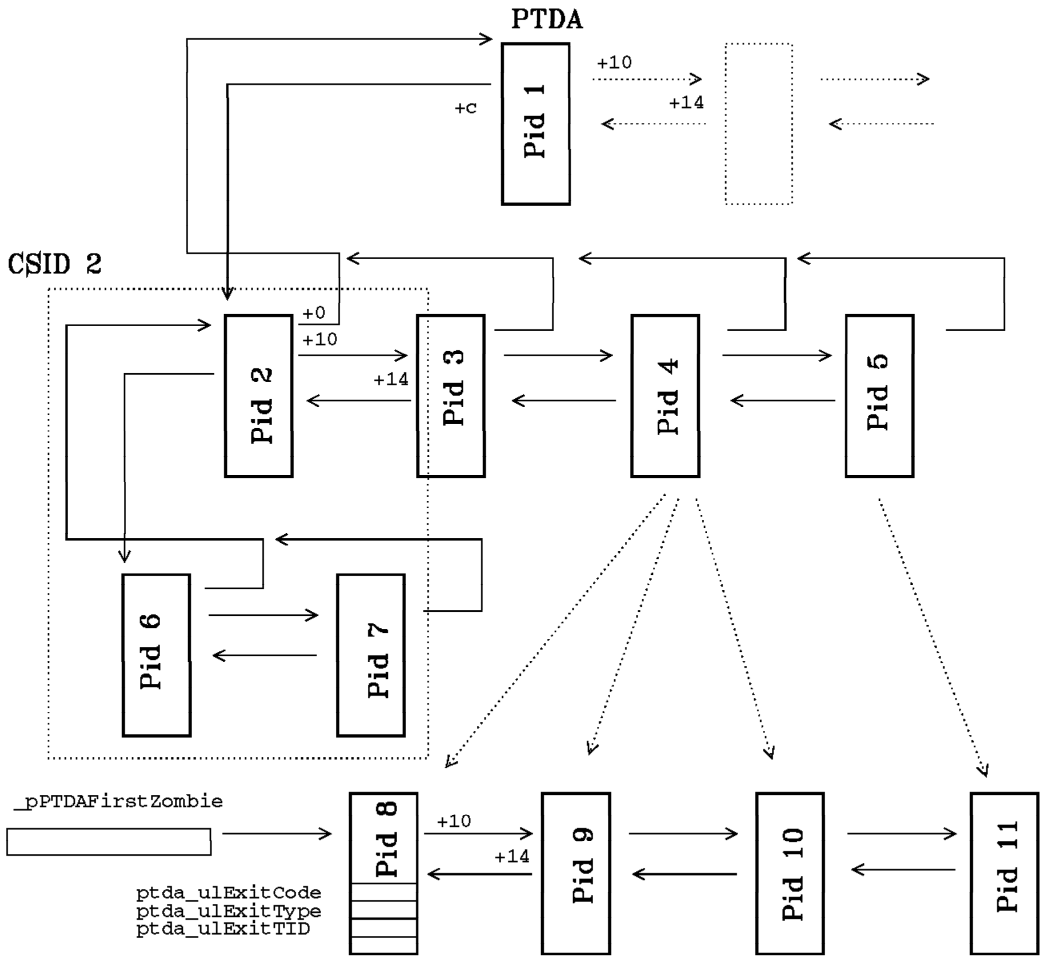

Thread Tree for a Process
PTDA
+20
ptda_pTCBCritSec
TCB
TSD
+14
Thread 1
TCB
TSD
+158
Thread 2
TCB
TSD
Thread 3
TCB
TSD
+15c
Thread 4
Thread 3 is in critical section
Thread 2 and Thread 1 are waiting for Thread 4 to die
RJM 28th Aug 95 - thdtree

Figure 23. Thread Tree for a Process

3.5.1.5 Process Trees, Subtrees and Zombies

The Process Tree, Subtrees and Zombies



Pid 1 is Detached (no parent)

Other Detached Processes are Siblings of Pid 1

Pids 1 - 7 are active

Pids 8 - 11 are dead (zombies)

Pid 4 may DosWaitChild on Pids 8 - 10

Pid 5 may DosWaitChild on Pid 11

RJM 28th Aug 95 - prtree

Figure 24. Process Trees, Subtrees and Zombies

3.5.1.6 Orphaned and Adopted Processes

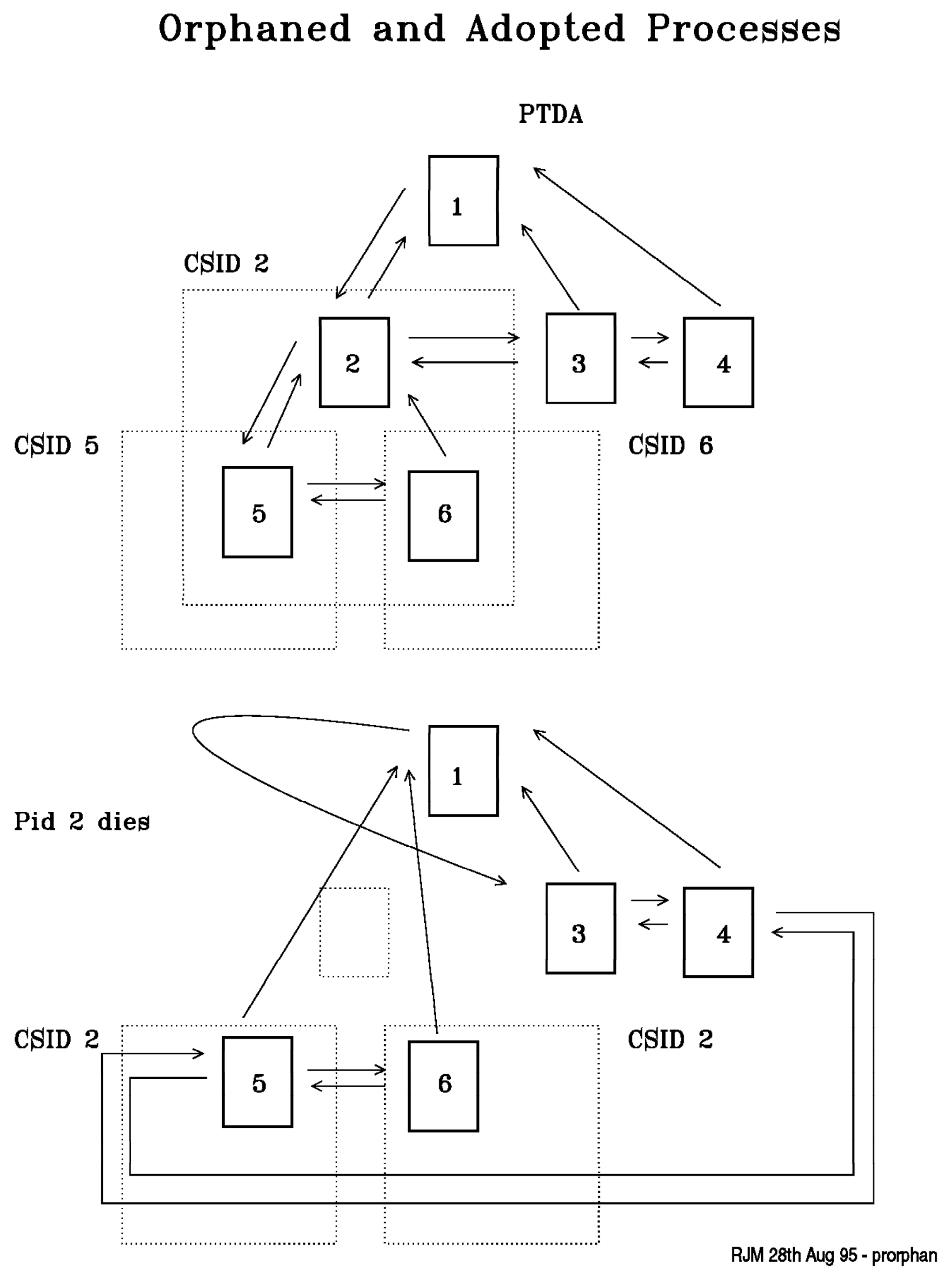
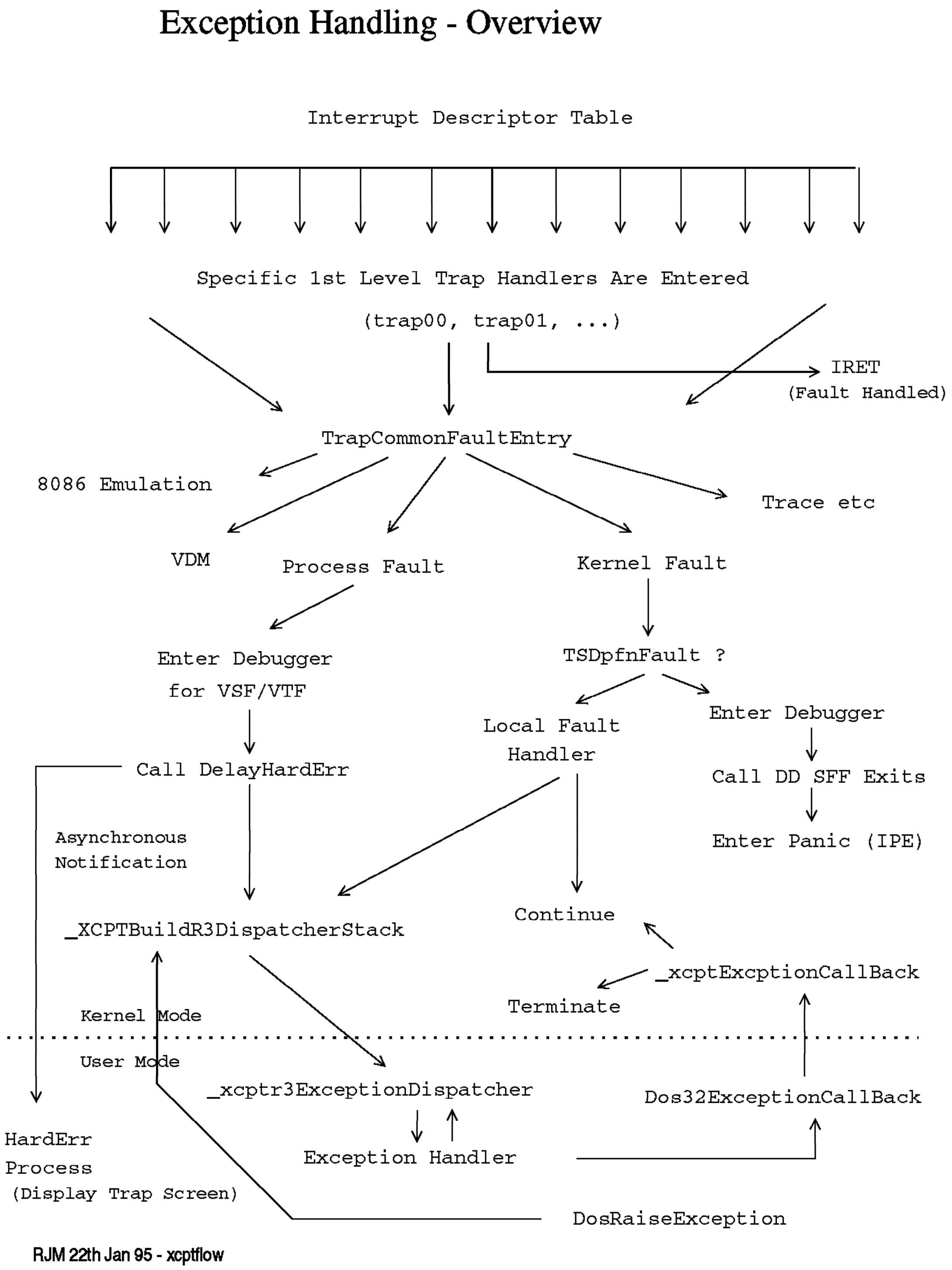


Figure 25. Orphaned and Adopted Processes

3.5.1.7 OS/2 Exception Management - Overview



RJM 22th Jan 95 - xcptflow

Figure 26. Exception Management Overview

3.5.1.8 Exception Handler Stack Frames

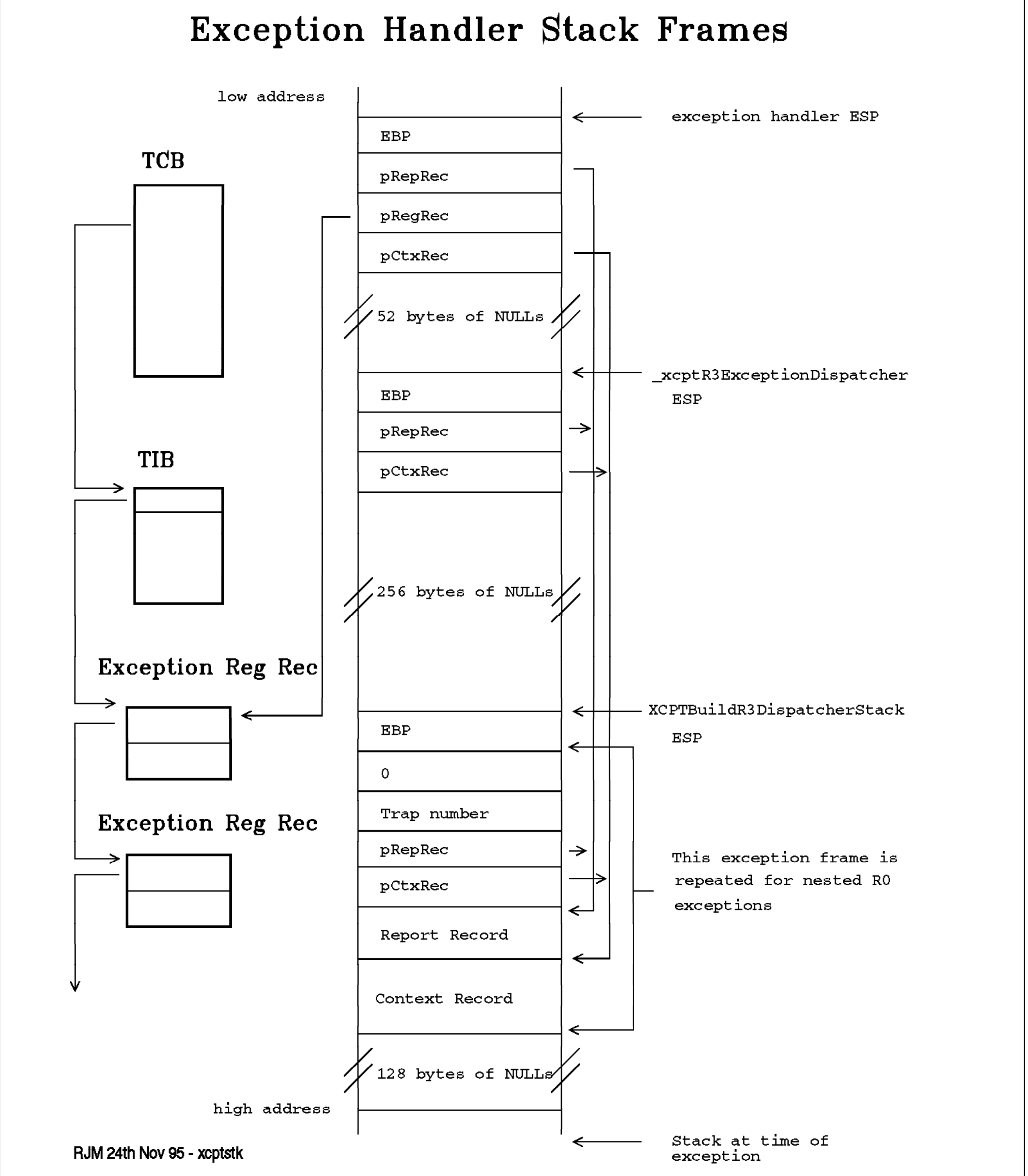


Figure 27. Exception Handler Stack Frames

3.5.2 Thread Control Block OS/2 Warp V3.0

For **TCB** formats for other versions of OS/2 see:

3.5.2.1, "Thread Control Block for OS/2 Warp V3.0 with FixPak" on page 105

3.5.2.2, "Thread Control Block for OS/2 Warp V3.0 with FixPak 11 or Later" on page 109

3.5.2.3, "Thread Control Block for OS/2 V2.11 with FixPak 90 or Later" on page 113

3.5.2.4, "Thread Control Block for OS/2 V2.11" on page 117

Pointers

_papTCBSlots points to the thread slot table of TCB pointers.

Multiple chain pointers between, TSD, TCB and PTDA.

CurrTCB points to the current TCB.

Locations

System Arena.

VM Owner

tcb (0xffcc)

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer

Field Name	Offset	Length	Type	Description
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBTLMA	+74	80	D	Thread local memory area
TCBDMAAdd	+f4	4	D	User′s I/O transfer address
TCBSecPos	+f8	4	D	Position of first sector accessed within file
TCBThisSFT	+fc	4	D	pointer to SFT we′re working with
TCBValSec	+100	4	D	Number of valid (previously written) sectors
TCBpRTCB	+104	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+108	2	W	process ID for file sharing checks
TCBUser_ID	+10a	2	W	user ID for file sharing checks
TCBfSharing	+10c	1	B	non-zero ==> no redirection
TCBSrvAttrib	+10d	1	B	see SetAttrib/file.asm
TCBJfnFlag	+10e	1	B	JFN flag bits for current file handle
TCBAllowed	+10f	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+110	4	D	server′s per file cookie
TCBOpFlags	+114	2	W	whether server wants oplock, etc.
TCBCurBuf	+116	4	D	currently assigned buffer
TCBThishVPB	+11a	2	W	handle of current VPB
TCBNextAdd	+11c	2	W	
TCBBytSecPos	+11e	2	W	position of first byte within sector
TCBClusNum	+120	2	W	

Field Name	Offset	Length	Type	Description
TCBLastPos	+122	2	W	
TCBBytCnt1	+124	2	W	Number of bytes in 1st sector
TCBBytCnt2	+126	2	W	# of bytes in last sector
TCBSecCnt	+128	2	W	number of whole sectors
TCBSecClusPos	+12a	1	B	posit of first sector within cluster
TCBBufHE	+12b	1	B	How to handle a HardError
TCBactBufHE	+12c	1	B	action response from user on HardErr
TCBfIOLock	+12d	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+12e	C	S	Lock handle of user mem
TCBThisCDS	+13a	4	D	Address of current CDS
TCBThisFSC	+13e	4	D	address of current FSC
TCBpTmpCDS	+142	4	D	Address of dummycds
TCBpOpenBuf	+146	2	W	Address of current OpenBuf
TCBpSearchBuf	+148	2	W	Address of SearchBuf
TCBFailErr	+14a	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+14c	4	D	16-bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+150	4	D	debugger display address for ksems
TCB_NPX_Buffer	+154	4	D	
TCBpTCBWaitNext	+158	4	D	Next waiting TCB
TCBpTCBWaitList	+15c	4	D	Threads waiting for me to die
TCBQState	+160	1	B	Scheduler queue location (actual)
TCBState	+161	1	B	Current scheduler state (desired)
TCBWakeFlags	+162	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+163	1	B	Window Boost count
TCBPriClass	+164	1	B	Priority Class (user)
TCBPriLevel	+165	1	B	Priority Level (user)
TCBPriClassMod	+166	1	B	Priority Class modifier bits
TCBSchFlags	+167	1	B	Misc. Scheduler flags
TCBPriority	+168	2	W	Calculated Priority
TCBPriorityMin	+16a	2	W	Minimum Scheduling priority
TCBcBoostLock	+16c	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+170	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+174	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+178	4	D	Higher priority thread

Field Name	Offset	Length	Type	Description
TCBpTCBPriLower	+17c	4	D	Lower priority thread
TCBpTCBPriNext	+180	4	D	Next same-priority thread
TCBpTCBPriPrev	+184	4	D	Prev same-priority thread
TCBpTCBWakeup	+188	4	D	TKQueryWakeup TCB list
TCBSleepID	+18c	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+190	10	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a0	1	B	Used by the loader
TCBfSwapping	+1a1	1	B	status of swapping
TCBVolIONest	+1a2	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+1a3	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1a4	4	D	I/O request packet for thread
TCBSysTime	+1a8	4	D	time spent in system code
TCBUserTime	+1ac	4	D	time spent in user code
TCB_pPVDBThd	+1b0	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+1b4	4	D	
TCBCpl2_ESP	+1b8	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1bc	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1be	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1bf	1	B	Kernel entry force flags
TCBSig_pend	+1c0	2	W	bit vector of pending signals
TCBSig_holding	+1c2	2	W	bit vector of postponed signals
TCBSig_cur	+1c4	2	W	bit vec of signals being processed
TCBXcptRepRec	+1c6	4	D	report record of active exception
TCBSig_termtid	+1ca	2	W	tid of terminator -75797
TCBSecbits	+1cc	1	B	Security bits 54735
TCBspbytes	+1cd	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1ce	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1d2	1	B	Used for hard error processing
TCBModeFlags	+1d3	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1d5	1	B	Spare flags
TCBLibiFlags	+1d6	1	B	84537

Field Name	Offset	Length	Type	Description
TCBFiller	+1d7	1	B	To keep size 4*N
TCB_ProcNameBuf	+1d8	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1dc	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e0	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1e4	4	D	Used by loader
TCB_FaultBuf	+1e8	4	D	
TCB_ObjNameBufL	+1ec	2	W	Length of object name buffer
TCB_TmpNameBufL	+1ee	2	W	
TCB_SrcModNameBufL	+1f0	2	W	
TCB_FaultBufL	+1f2	2	W	
TCBSecchild	+1f4	4	D	Child Security data 54735

Table 61. TCBForcedActions Flag Definitions

Name	Bit Mask	Description
TK_FF_BUF	0x00000001	Buffer must be released
TK_FF_EXIT	0x00000002	Call TKExit (old FF_DES)
TK_FF_CRITSEC	0x00000004	Enter Per-task critical section
TK_FF_ICE	0x00000008	Freeze thread
TK_FF_NPX	0x00000010	NPX Error
TK_FF_TIB	0x00000020	Update the TIB
TK_FF_TRC	0x00000040	Enter Debug
TK_FF_SIG	0x00000080	Signal pending
TK_FF_CTXH	0x00000100	Pending local context hooks
TK_FF_STIH	0x00000200	Execute STI hooks
TK_FF_VDMBP	0x00000400	Execute VDM BP hooks
TK_FF_RTRY	0x00000800	Retry V86 system call
TK_FF_PIB	0x00001000	Update the PIB
TK_FF_SCH	0x00002000	Do Scheduler Processing
TK_FF_TFBIT	0x00004000	Validate user eflags TF bit
TK_FF_TIBPRI	0x00008000	Update only the priority fields in TIB 59463

Table 62. TCBEntryActions Flag Definitions

Name	Bit Mask	Description
TK_EF_PFCLI	1	Page fault inside CLI
TK_EF_TRC	2	DosDebug action pending

Table 63. TCBWakeFlags Flag Definitions

Name	Bit Mask	Description
TK_WF_INTERRUPTED	0x01	Sleep was interrupted
TK_WF_TIMEEXP	0x02	Timeout expired
TK_WF_INTPENDING	0x04	Interrupt pending
TK_WF_SINGLEWAKEUP	0x08	Thread wants single wakeup
TK_WF_INTERRUPTIBLE	0x10	Thread blocked interruptibly
TK_WF_TIMEOUT	0x20	Thread blocked with timeout
TK_WF_SLEEPING	0x40	In TKSleep()

Table 64. TCBState and TCBQState Definitions

Name	Value	Description
STATE_VOID	0	Uninitialized
STATE_READY	1	Ready to run
STATE_BLOCKED	2	Blocked on an ID
STATE_SUSPENDED	3	Suspended (DosSuspendThread)
STATE_CRITSEC	4	Blocked by another CritSec thread
STATE_RUNNING	5	Thread currently running
STATE_READYBOOST	6	Ready, but apply an IO boost
STATE_TSD	7	Thread waiting for TSD
STATE_DELAYED	8	Delayed TKWakeup (Almost Ready)
STATE_FROZEN	9	Frozen Thread (FF_ICE)
STATE_GETSTACK	10	Incoming TSD
STATE_BADSTACK	11	TSD failed to swap in

Table 65. TCBPriClassMod Definitions

Name	Value	Description
CLASSMOD_KEYBOARD	0x04	Keyboard boost
CLASSMOD_STARVED	0x08	Starvation boost
CLASSMOD_DEVICE	0x10	Device I/O Done Boost
CLASSMOD_FOREGROUND	0x20	Foreground boost
CLASSMOD_WINDOW	0x40	Window Boost
CLASSMOD_VDM_INTERRUPT	0x80	VDM simulated interrupt boost

Table 66 (Page 1 of 2). TCBPriClass Definitions

Name	Value	Description
CLASS_NOCHANGE	0x00	No priority class change
CLASS_IDLE_TIME	0x01	Idle-Time class
CLASS_REGULAR	0x02	Regular class
CLASS_TIME_CRITICAL	0x03	Time-Critical class

Table 66 (Page 2 of 2). TCBPriClass Definitions		
Name	**Value**	**Description**
CLASS_SERVER	0x04	Client/Server Server class

Table 67. TCBSchFlg Flag Definitions		
Name	**Bit Mask**	**Description**
SCH_PROTECTED_PRI	0x0001	Only Intra-process SetPri allowed
SCH_WINDOWBOOST_LOCK	0x0002	Lock out windoboost changes
SCH_MINSLICE	0x0004	Use minimum timeslice
SCH_PAGE_FAULT	0x0008	Dynamic timeslicing ###
SCH_PAGE_FAULT_BIT	0x03	Dynamic timeslicing P728371

Table 68. TCBfSwapping Flag Definitions		
Name	**Bit Mask**	**Description**
SM_TCB_SWAPPING	0x01	swap I/O underway
SM_TCB_RESIZING	0x02	data structures are growing

Table 69. TCBMiscFlags Flag Definitions		
Name	**Bit Mask**	**Description**
TMF_CMapFailed	(0x01)	Set if alloc/realloc failed on a cluster map (mft_selCMap).
TMF_IGNORE_HE	(0x02)	If set, ignore (auto fail) hard error
TMF_MULT_XCPT	(0x04)	Set if multiple ring 0 exceptions
TMF_NoFwd	(0x08)	Set if inhibiting forwarders
TMF_EXIT_TERM	(0x10)	TK_FF_EXIT means TKTermThread
TMF_NO_EXCEPT	(0x20)	Indicates TIB exception field invalid
TMF_XCPT_HE	(0x40)	Indicates an exception harderr is pending

Table 70. TCBMSpareFlags Flag Definitions		
Name	**Bit Mask**	**Description**
SPFLAGS_FGND_DISKIO	0x0080	Foreground Disk I/O

Table 71. TCBReqPktFlg Flag Definitions		
Name	**Bit Mask**	**Description**
TK_RP_ALLOCATED	0x01	
TK_RP_INUSE	0x02	

3.5.2.1 Thread Control Block for OS/2 Warp V3.0 with FixPak

See FixPak 09 for details of the changes introduced in this FixPak.

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBTLMA	+78	80	D	Thread local memory area
TCBDMAAdd	+f8	4	D	User's I/O transfer address

Field Name	Offset	Length	Type	Description
TCBSecPos	+fc	4	D	Position of first sector accessed within file
TCBThisSFT	+100	4	D	pointer to SFT we′re working with
TCBValSec	+104	4	D	Number of valid (previously written) sectors
TCBpRTCB	+108	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+10c	2	W	process ID for file sharing checks
TCBUser_ID	+10e	2	W	user ID for file sharing checks
TCBfSharing	+110	1	B	non-zero ==> no redirection
TCBSrvAttrib	+111	1	B	see SetAttrib/file.asm
TCBJfnFlag	+112	1	B	JFN flag bits for current fil handle
TCBAllowed	+113	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+114	4	D	server′s per file cookie
TCBOpFlags	+118	2	W	whether server wants oplock, etc.
TCBCurBuf	+11a	4	D	currently assigned buffer
TCBThishVPB	+11e	2	W	handle of current VPB
TCBNextAdd	+120	2	W	
TCBBytSecPos	+122	2	W	position of first byte within sector
TCBClusNum	+124	2	W	
TCBLastPos	+126	2	W	
TCBBytCnt1	+128	2	W	Number of bytes in 1st sector
TCBBytCnt2	+12a	2	W	# of bytes in last sector
TCBSecCnt	+12c	2	W	number of whole sectors
TCBSecClusPos	+12e	1	B	posit of first sector within cluster
TCBBufHE	+12f	1	B	How to handle a HardError
TCBactBufHE	+130	1	B	action response from user on HardErr
TCBfIOLock	+131	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+132	C	S	Lock handle of user mem
TCBThisCDS	+13e	4	D	Address of current CDS
TCBThisFSC	+142	4	D	address of current FSC
TCBpTmpCDS	+146	4	D	Address of dummycds
TCBpOpenBuf	+14a	2	W	Address of current OpenBuf
TCBpSearchBuf	+14c	2	W	Address of SearchBuf

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBFailErr	+14e	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+150	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+154	4	D	debugger display address for ksems
TCB_NPX_Buffer	+158	4	D	
TCBpTCBWaitNext	+15c	4	D	Next waiting TCB
TCBpTCBWaitList	+160	4	D	Threads waiting for me to die
TCBQState	+164	1	B	Scheduler queue location (actual)
TCBState	+165	1	B	Current scheduler state (desired)
TCBWakeFlags	+166	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+167	1	B	Window Boost count
TCBPriClass	+168	1	B	Priority Class (user)
TCBPriLevel	+169	1	B	Priority Level (user)
TCBPriClassMod	+16a	1	B	Priority Class modifier bits
TCBSchFlags	+16b	1	B	Misc. Scheduler flags
TCBPriority	+16c	2	W	Calculated Priority
TCBPriorityMin	+16e	2	W	Minimum Scheduling priority
TCBcBoostLock	+170	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+174	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+178	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+17c	4	D	Higher priority thread
TCBpTCBPriLower	+180	4	D	Lower priority thread
TCBpTCBPriNext	+184	4	D	Next same-priority thread
TCBpTCBPriPrev	+188	4	D	Prev same-priority thread
TCBpTCBWakeup	+18c	4	D	TKQueryWakeup TCB list
TCBSleepID	+190	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+194	10	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a4	1	B	Used by the loader
TCBfSwapping	+1a5	1	B	status of swapping
TCBVolIONest	+1a6	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+1a7	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1a8	4	D	I/O request packet for thread
TCBSysTime	+1ac	4	D	time spent in system code
TCBUserTime	+1b0	4	D	time spent in user code

Field Name	Offset	Length	Type	Description
TCB_pPVDBThd	+1b4	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+1b8	4	D	
TCBCpl2_ESP	+1bc	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1c0	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1c2	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1c3	1	B	Kernel entry force flags
TCBSig_pend	+1c4	2	W	bit vector of pending signals
TCBSig_holding	+1c6	2	W	bit vector of postponed signals
TCBSig_cur	+1c8	2	W	bit vec of signals being processed
TCBXcptRepRec	+1ca	4	D	report record of active exception
TCBSig_termtid	+1ce	2	W	tid of terminator -75797
TCBSecbits	+1d0	1	B	Security bits 54735
TCBspbytes	+1d1	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1d2	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1d6	1	B	Used for hard error processing
TCBModeFlags	+1d7	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1d9	1	B	Spare flags
TCBLibiFlags	+1da	1	B	84537
TCBFiller	+1db	1	B	To keep size 4*N
TCB_ProcNameBuf	+1dc	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1e0	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e4	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1e8	4	D	Used by loader
TCB_FaultBuf	+1ec	4	D	
TCB_ObjNameBufL	+1f0	2	W	Length of object name buffer
TCB_TmpNameBufL	+1f2	2	W	
TCB_SrcModNameBufL	+1f4	2	W	
TCB_FaultBufL	+1f6	2	W	
TCBSecchild	+1f8	4	D	Child Security data 54735

Table 72. TCBLibiFlags Flag Definitions

Name	Bit Mask	Description
INIT_ROUTINE_FAILED	(0x01)	84537 Set if dll init routine failed

3.5.2.2 Thread Control Block for OS/2 Warp V3.0 with FixPak 11 or Later

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value

Field Name	Offset	Length	Type	Description
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBTLMA	+78	80	D	Thread local memory area
TCBDMAAdd	+f8	4	D	User′s I/O transfer address
TCBSecPos	+fc	4	D	Position of first sector accessed within file
TCBThisSFT	+100	4	D	pointer to SFT we′re working with
TCBValSec	+104	4	D	Number of valid (previously written) sectors
TCBpRTCB	+108	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+10c	2	W	process ID for file sharing checks
TCBUser_ID	+10e	2	W	user ID for file sharing checks
TCBfSharing	+110	1	B	non-zero ==> no redirection
TCBSrvAttrib	+111	1	B	see SetAttrib/file.asm
TCBJfnFlag	+112	1	B	JFN flag bits for current fil handle
TCBAllowed	+113	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+114	4	D	server′s per file cookie
TCBOpFlags	+118	2	W	whether server wants oplock, etc.
TCBCurBuf	+11a	4	D	currently assigned buffer
TCBThishVPB	+11e	2	W	handle of current VPB
TCBNextAdd	+120	2	W	
TCBBytSecPos	+122	2	W	position of first byte within sector
TCBClusNum	+124	2	W	
TCBLastPos	+126	2	W	
TCBBytCnt1	+128	2	W	Number of bytes in 1st sector
TCBBytCnt2	+12a	2	W	# of bytes in last sector
TCBSecCnt	+12c	2	W	number of whole sectors
TCBSecClusPos	+12e	1	B	posit of first sector within cluster
TCBBufHE	+12f	1	B	How to handle a HardError
TCBactBufHE	+130	1	B	action response from user on HardErr
TCBfIOLock	+131	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+132	C	S	Lock handle of user mem

Field Name	Offset	Length	Type	Description
TCBThisCDS	+13e	4	D	Address of current CDS
TCBThisFSC	+142	4	D	address of current FSC
TCBpTmpCDS	+146	4	D	Address of dummycds
TCBpOpenBuf	+14a	2	W	Address of current OpenBuf
TCBpSearchBuf	+14c	2	W	Address of SearchBuf
TCBFailErr	+14e	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+150	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+154	4	D	debugger display address for ksems
TCB_NPX_Buffer	+158	4	D	
TCBpTCBWaitNext	+15c	4	D	Next waiting TCB
TCBpTCBWaitList	+160	4	D	Threads waiting for me to die
TCBQState	+164	1	B	Scheduler queue location (actual)
TCBState	+165	1	B	Current scheduler state (desired)
TCBWakeFlags	+166	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+167	1	B	Window Boost count
TCBPriClass	+168	1	B	Priority Class (user)
TCBPriLevel	+169	1	B	Priority Level (user)
TCBPriClassMod	+16a	1	B	Priority Class modifier bits
TCBSchFlags	+16b	1	B	Misc. Scheduler flags
TCBPriority	+16c	2	W	Calculated Priority
TCBPriorityMin	+16e	2	W	Minimum Scheduling priority
TCBcBoostLock	+170	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+174	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+178	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+17c	4	D	Higher priority thread
TCBpTCBPriLower	+180	4	D	Lower priority thread
TCBpTCBPriNext	+184	4	D	Next same-priority thread
TCBpTCBPriPrev	+188	4	D	Prev same-priority thread
TCBpTCBWakeup	+18c	4	D	TKQueryWakeup TCB list
TCBSleepID	+190	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+194	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a8	1	B	Used by the loader
TCBfSwapping	+1a9	1	B	status of swapping
TCBVolIONest	+1aa	1	B	nesting level of FSH_DoVolIO

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBReqPktFlg	+1ab	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1ac	4	D	I/O request packet for thread
TCBSysTime	+1b0	4	D	time spent in system code
TCBUserTime	+1b4	4	D	time spent in user code
TCB_pPVDBThd	+1b8	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+1bc	4	D	
TCBCpl2_ESP	+1c0	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1c4	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1c6	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1c7	1	B	Kernel entry force flags
TCBSig_pend	+1c8	2	W	bit vector of pending signals
TCBSig_holding	+1ca	2	W	bit vector of postponed signals
TCBSig_cur	+1cc	2	W	bit vec of signals being processed
TCBXcptRepRec	+1ce	4	D	report record of active exception
TCBSig_termtid	+1d2	2	W	tid of terminator -75797
TCBSecbits	+1d4	1	B	Security bits 54735
TCBspbytes	+1d5	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1d6	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1da	1	B	Used for hard error processing
TCBModeFlags	+1db	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1dd	1	B	Spare flags
TCBLibiFlags	+1de	1	B	84537
TCBFiller	+1df	1	B	To keep size 4*N
TCB_ProcNameBuf	+1e0	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1e4	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e8	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1ec	4	D	Used by loader
TCB_FaultBuf	+1f0	4	D	
TCB_ObjNameBufL	+1f4	2	W	Length of object name buffer
TCB_TmpNameBufL	+1f6	2	W	
TCB_SrcModNameBufL	+1f8	2	W	

Field Name	Offset	Length	Type	Description
TCB_FaultBufL	+1fa	2	W	
TCBSecchild	+1fc	4	D	Child Security data 54735

3.5.2.3 Thread Control Block for OS/2 V2.11 with FixPak 90 or Later

Feature 82818 introduces the Kernel Debugger `.MK` command. 82818 is supplied as an APAR fix to:

OS/2 Warp V3.0 as PJ18364 in FixPak 7.

OS/2 V2.11 as PJ16805 in FixPak 90.

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD

Field Name	Offset	Length	Type	Description
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBDMAAdd	+78	4	D	User′s I/O transfer address
TCBSecPos	+7c	4	D	Position of first sector accessed within file
TCBThisSFT	+80	4	D	pointer to SFT we′re working with
TCBValSec	+84	4	D	Number of valid (previously written) sectors
TCBpRTCB	+88	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+8c	2	W	process ID for file sharing checks
TCBUser_ID	+8e	2	W	user ID for file sharing checks
TCBfSharing	+90	1	B	non-zero ==> no redirection
TCBSrvAttrib	+91	1	B	see SetAttrib/file.asm
TCBJfnFlag	+92	1	B	JFN flag bits for current fil handle
TCBAllowed	+93	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+94	4	D	server′s per file cookie
TCBOpFlags	+98	2	W	whether server wants oplock, etc.
TCBCurBuf	+9a	4	D	currently assigned buffer
TCBThishVPB	+9e	2	W	handle of current VPB
TCBNextAdd	+a0	2	W	
TCBBytSecPos	+a2	2	W	position of first byte within sector
TCBClusNum	+a4	2	W	
TCBLastPos	+a6	2	W	
TCBBytCnt1	+a8	2	W	Number of bytes in 1st sector
TCBBytCnt2	+aa	2	W	# of bytes in last sector
TCBSecCnt	+ac	2	W	number of whole sectors
TCBSecClusPos	+ae	1	B	posit of first sector within cluster
TCBBufHE	+af	1	B	How to handle a HardError
TCBactBufHE	+b0	1	B	action response from user on HardErr
TCBfIOLock	+b1	1	B	NZ if TCBLockHndl is valid

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBLockHndl	+b2	C	S	Lock handle of user mem
TCBThisCDS	+be	4	D	Address of current CDS
TCBThisFSC	+c2	4	D	address of current FSC
TCBpTmpCDS	+c6	4	D	Address of dummycds
TCBpOpenBuf	+ca	2	W	Address of current OpenBuf
TCBpSearchBuf	+cc	2	W	Address of SearchBuf
TCBFailErr	+ce	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+d0	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+d4	4	D	debugger display address for ksems
TCB_NPX_Buffer	+d8	4	D	
TCBpTCBWaitNext	+dc	4	D	Next waiting TCB
TCBpTCBWaitList	+e0	4	D	Threads waiting for me to die
TCBQState	+e4	1	B	Scheduler queue location (actual)
TCBState	+e5	1	B	Current scheduler state (desired)
TCBWakeFlags	+e6	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+e7	1	B	Window Boost count
TCBPriClass	+e8	1	B	Priority Class (user)
TCBPriLevel	+e9	1	B	Priority Level (user)
TCBPriClassMod	+ea	1	B	Priority Class modifier bits
TCBSchFlags	+eb	1	B	Misc. Scheduler flags
TCBPriority	+ec	2	W	Calculated Priority
TCBPriorityMin	+ee	2	W	Minimum Scheduling priority
TCBcBoostLock	+f0	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+f4	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+f8	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+fc	4	D	Higher priority thread
TCBpTCBPriLower	+100	4	D	Lower priority thread
TCBpTCBPriNext	+104	4	D	Next same-priority thread
TCBpTCBPriPrev	+108	4	D	Prev same-priority thread
TCBpTCBWakeup	+10c	4	D	TKQueryWakeup TCB list
TCBSleepID	+110	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+114	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+128	1	B	Used by the loader
TCBfSwapping	+129	1	B	status of swapping

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBVolIONest	+12a	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+12b	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+12c	4	D	I/O request packet for thread
TCBpMemStatCur	+130	4	D	Current structure being filled in
TCBMemStat	+134	3C	S	statistics structure
TCBSysTime	+170	4	D	time spent in system code
TCBUserTime	+174	4	D	time spent in user code
TCB_pPVDBThd	+178	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+17c	4	D	
TCBCpl2_ESP	+180	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+184	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+186	1	W	Value copied from ptda_NewFiles
TCBEntryActions	+187	1	B	Kernel entry force flags
TCBSig_pend	+188	2	W	bit vector of pending signals
TCBSig_holding	+18a	2	W	bit vector of postponed signals
TCBSig_cur	+18c	2	W	bit vec of signals being processed
TCBXcptRepRec	+18e	4	D	report record of active exception
TCBSig_termtid	+192	2	W	
TCBSecbits	+194	1	B	Security bits 54735
TCBspbytes	+195	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+196	4	D	
TCBMiscFlags	+19a	1	D	Used for hard error processing
TCBModeFlags	+19b	2	D	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+19d	1	B	Spare flags
TCBLibiFlags	+19e	1	B	
TCBFiller	+19f	1	B	
TCB_ProcNameBuf	+1a0	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1a4	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1a8	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1ac	4	D	Used by loader
TCB_FaultBuf	+1b0	4	D	

Field Name	Offset	Length	Type	Description
TCB_ObjNameBufL	+1b4	2	W	Length of object name buffer
TCB_TmpNameBufL	+1b6	2	W	
TCB_SrcModNameBufL	+1b8	2	W	
TCB_FaultBufL	+1ba	2	W	
TCBSecchild	+1bc	4	D	Child Security data 54735

Table 73. TCBLibiFlags Flag Definitions

Name	Bit Mask	Description
INIT_ROUTINE_FAILED	(0x01)	84537 Set if dll init routine failed

3.5.2.4 Thread Control Block for OS/2 V2.11

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA

Field Name	**Offset**	**Length**	**Type**	**Description**
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBDMAAdd	+74	4	D	User′s I/O transfer address
TCBSecPos	+78	4	D	Position of first sector accessed within file
TCBThisSFT	+7c	4	D	pointer to SFT we′re working with
TCBValSec	+80	4	D	Number of valid (previously written) sectors
TCBpRTCB	+84	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+88	2	W	process ID for file sharing checks
TCBUser_ID	+8a	2	W	user ID for file sharing checks
TCBfSharing	+8c	1	B	non-zero ==> no redirection
TCBSrvAttrib	+8d	1	B	see SetAttrib/file.asm
TCBJfnFlag	+8e	1	B	JFN flag bits for current fil handle
TCBAllowed	+8f	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+90	4	D	server′s per file cookie
TCBOpFlags	+94	2	W	whether server wants oplock, etc.
TCBCurBuf	+96	4	D	currently assigned buffer
TCBThishVPB	+9a	2	W	handle of current VPB
TCBNextAdd	+9c	2	W	
TCBBytSecPos	+9e	2	W	position of first byte within sector
TCBClusNum	+a0	2	W	
TCBLastPos	+a2	2	W	
TCBBytCnt1	+a4	2	W	Number of bytes in 1st sector
TCBBytCnt2	+a6	2	W	# of bytes in last sector
TCBSecCnt	+a8	2	W	number of whole sectors
TCBSecClusPos	+aa	1	B	posit of first sector within cluster

Field Name	Offset	Length	Type	Description
TCBBufHE	+ab	1	B	How to handle a HardError
TCBactBufHE	+ac	1	B	action response from user on HardErr
TCBfIOLock	+ad	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+ae	C	S	Lock handle of user mem
TCBThisCDS	+ba	4	D	Address of current CDS
TCBThisFSC	+be	4	D	address of current FSC
TCBpTmpCDS	+c2	4	D	Address of dummycds
TCBpOpenBuf	+c6	2	W	Address of current OpenBuf
TCBpSearchBuf	+c8	2	W	Address of SearchBuf
TCBFailErr	+ca	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+cc	4	D	16-bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+d0	4	D	debugger display address for ksems
TCB_NPX_Buffer	+d4	4	D	
TCBpTCBWaitNext	+d8	4	D	Next waiting TCB
TCBpTCBWaitList	+dc	4	D	Threads waiting for me to die
TCBQState	+e0	1	B	Scheduler queue location (actual)
TCBState	+e1	1	B	Current scheduler state (desired)
TCBWakeFlags	+e2	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+e3	1	B	Window Boost count
TCBPriClass	+e4	1	B	Priority Class (user)
TCBPriLevel	+e5	1	B	Priority Level (user)
TCBPriClassMod	+e6	1	B	Priority Class modifier bits
TCBSchFlags	+e7	1	B	Misc. Scheduler flags
TCBPriority	+e8	2	W	Calculated Priority
TCBPriorityMin	+ea	2	W	Minimum Scheduling priority
TCBcBoostLock	+ec	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+f0	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+f4	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+f8	4	D	Higher priority thread
TCBpTCBPriLower	+fc	4	D	Lower priority thread
TCBpTCBPriNext	+100	4	D	Next same-priority thread
TCBpTCBPriPrev	+104	4	D	Prev same-priority thread
TCBpTCBWakeup	+108	4	D	TKQueryWakeup TCB list
TCBSleepID	+10c	4	D	Sleep ID this TCB is sleeping on

Field Name	Offset	Length	Type	Description
TCBtoe	+110	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+124	1	B	Used by the loader
TCBfSwapping	+125	1	B	status of swapping
TCBVolIONest	+126	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+127	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+128	4	D	I/O request packet for thread
TCBpMemStatCur	+12c	4	D	Current structure being filled in
TCBMemStat	+130	3C	S	statistics structure
TCBSysTime	+16c	4	D	time spent in system code
TCBUserTime	+170	4	D	time spent in user code
TCB_pPVDBThd	+174	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+178	4	D	
TCBCpl2_ESP	+17c	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+180	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+182	1	W	Value copied from ptda_NewFiles
TCBEntryActions	+183	1	B	Kernel entry force flags
TCBSig_pend	+184	2	W	bit vector of pending signals
TCBSig_holding	+186	2	W	bit vector of postponed signals
TCBSig_cur	+188	2	W	bit vec of signals being processed
TCBXcptRepRec	+18a	4	D	report record of active exception
TCBSig_termtid	+18e	2	W	
TCBSecbits	+190	1	B	Security bits 54735
TCBspbytes	+191	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+192	4	D	
TCBMiscFlags	+196	1	D	Used for hard error processing
TCBModeFlags	+197	2	D	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+199	1	B	Spare flags
TCBLibiFlags	+19a	1	B	
TCBFiller	+19b	1	B	
TCB_ProcNameBuf	+19c	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1a0	4	D	Pointer to object name buffer

Field Name	Offset	Length	Type	Description
TCB_TmpNameBuf	+1a4	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1a8	4	D	Used by loader
TCB_FaultBuf	+1ac	4	D	
TCB_ObjNameBufL	+1b0	2	W	Length of object name buffer
TCB_TmpNameBufL	+1b2	2	W	
TCB_SrcModNameBufL	+1b4	2	W	
TCB_FaultBufL	+1b6	2	W	
TCBSecchild	+1b8	4	D	Child Security data 54735

3.5.3 Thread Swappable Data for OS/2 Warp V3.0 ALLSTRICT Kernel

For **TSD** formats for other versions of OS/2 see:

3.5.3.1, “Thread Swappable Data for OS/2 Warp V3.0 RETAIL Kernel” on page 122

3.5.3.2, “Thread Swappable Data for OS/2 V2.11 ALLSTRICT Kernel” on page 123

3.5.3.3, “Thread Swappable Data for OS/2 V2.11 RETAIL Kernel” on page 124

Pointers

TCBpTSD points to the TSD associated with a TCB

CurrTSD points to the current TSD.

Locations

System Arena.

VM Owner

tsd (0xffcd)

Format

Field Name	Offset	Length	Type	Description
TSDpad	+0	1000	B	Dummy page to catch faults
TSDUserStack	+1000	F98	W	Thread′s kernel stack
TSDUserESP	+1f98	4	D	Saved user stack pointer
TSDUserSS	+1f9c	2	W	Saved user stack segment
TSDUserSSPad	+1f9e	2	W	Pad word pushed by gate
TSDKernelESP	+1fa0	4	D	Saved kernel stack pointer.
TSDpTCB	+1fa4	4	D	Link to TCB
TSDpfnFault	+1fa8	4	D	ptr to local fault handler in effect
TSDTrapNum	+1fac	4	D	TrapNum from the last fault
TSDerrcFault	+1fb0	4	D	error code from the last fault

Field Name	Offset	Length	Type	Description
TSDpljmp	+1fb4	4	D	Buffer saved by TKCatchFault
TSDselFault	+1fb8	2	W	faulting selector
TSDCpl2_SSSize	+1fba	2	W	Size of ring 2 stack - at least that's what the user believes
TSDdescLDT	+1fbc	8	D	LDT table descriptor
TSDdescKStackSS	+1fc4	8	D	SS descriptor
TSDdescFPEM	+1fcc	8	D	reserved descriptor slot
TSDdescTIB	+1fd4	8	D	FS mapping to TIB
TSDulExitCode	+1fdc	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+1fe0	4	D	error id from page fault
TSDPFErr	+1fe4	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+1fe8	4	D	
TSDlDbgRangeEnd	+1fec	4	D	
TSDlDbgLastAddr	+1ff0	4	D	
TSDpPCB	+1ff4	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+1ff8	4	D	Pointer to data buffer
TSDcObjSem	+1ffc	4	D	Count of object semaphores held

3.5.3.1 Thread Swappable Data for OS/2 Warp V3.0 RETAIL Kernel

Field Name	Offset	Length	Type	Description
TSDUserStack	+0	F9C	W	Thread's kernel stack
TSDUserESP	+f9c	4	D	Saved user stack pointer
TSDUserSS	+fa0	2	W	Saved user stack segment
TSDUserSSPad	+fa2	2	W	Pad word pushed by gate
TSDKernelESP	+fa4	4	D	Saved kernel stack pointer.
TSDpTCB	+fa8	4	D	Link to TCB
TSDpfnFault	+fac	4	D	ptr to local fault handler in effect
TSDTrapNum	+fb0	4	D	TrapNum from the last fault
TSDerrcFault	+fb4	4	D	error code from the last fault
TSDpljmp	+fb8	4	D	Buffer saved by TKCatchFault
TSDselFault	+fbc	2	W	faulting selector
TSDCpl2_SSSize	+fbe	2	W	Size of ring 2 stack - at least that's what the user believes

Field Name	Offset	Length	Type	Description
TSDdescLDT	+fc0	8	D	LDT table descriptor
TSDdescKStackSS	+fc8	8	D	SS descriptor
TSDdescFPEM	+fd0	8	D	reserved descriptor slot
TSDdescTIB	+fd8	8	D	FS mapping to TIB
TSDulExitCode	+fe0	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+fe4	4	D	error id from page fault
TSDPFErr	+fe8	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+fec	4	D	
TSDlDbgRangeEnd	+ff0	4	D	
TSDlDbgLastAddr	+ff4	4	D	
TSDpPCB	+ff8	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+ffc	4	D	Pointer to data buffer

3.5.3.2 Thread Swappable Data for OS/2 V2.11 ALLSTRICT Kernel

Field Name	Offset	Length	Type	Description
TSDpad	+0	1000	B	Dummy page to catch faults
TSDUserStack	+1000	F98	W	Thread′s kernel stack
TSDUserESP	+1f98	4	D	Saved user stack pointer
TSDUserSS	+1f9c	2	W	Saved user stack segment
TSDUserSSPad	+1f9e	2	W	Pad word pushed by gate
TSDKernelESP	+1fa0	4	D	Saved kernel stack pointer.
TSDpTCB	+1fa4	4	D	Link to TCB
TSDpfnFault	+1fa8	4	D	ptr to local fault handler in effect
TSDTrapNum	+1fac	4	D	TrapNum from the last fault
TSDerrcFault	+1fb0	4	D	error code from the last fault
TSDpljmp	+1fb4	4	D	Buffer saved by TKCatchFault
TSDselFault	+1fb8	2	W	faulting selector
TSDCpl2_SSSize	+1fba	2	W	Size of ring 2 stack - at least that′s what the user believes
TSDdescLDT	+1fbc	8	D	LDT table descriptor
TSDdescKStackSS	+1fc4	8	D	SS descriptor
TSDdescFPEM	+1fcc	8	D	reserved descriptor slot
TSDdescTIB	+1fd4	8	D	FS mapping to TIB
TSDulExitCode	+1fdc	4	D	Proposed Thread Exit code (for dbg)

Field Name	Offset	Length	Type	Description
TSDerridFault	+1fe0	4	D	error id from page fault
TSDPFErr	+1fe4	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+1fe8	4	D	
TSDlDbgRangeEnd	+1fec	4	D	
TSDlDbgLastAddr	+1ff0	4	D	
TSDpPCB	+1ff4	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+1ff8	4	D	Pointer to data buffer
TSDcObjSem	+1ffc	4	D	Count of object semaphores held

3.5.3.3 Thread Swappable Data for OS/2 V2.11 RETAIL Kernel

Field Name	Offset	Length	Type	Description
TSDUserStack	+0	F9C	W	Thread′s kernel stack
TSDUserESP	+f9c	4	D	Saved user stack pointer
TSDUserSS	+fa0	2	W	Saved user stack segment
TSDUserSSPad	+fa2	2	W	Pad word pushed by gate
TSDKernelESP	+fa4	4	D	Saved kernel stack pointer.
TSDpTCB	+fa8	4	D	Link to TCB
TSDpfnFault	+fac	4	D	ptr to local fault handler in effect
TSDTrapNum	+fb0	4	D	TrapNum from the last fault
TSDerrcFault	+fb4	4	D	error code from the last fault
TSDpljmp	+fb8	4	D	Buffer saved by TKCatchFault
TSDselFault	+fbc	2	W	faulting selector
TSDCpl2_SSSize	+fbe	2	W	Size of ring 2 stack - at least that′s what the user believes
TSDdescLDT	+fc0	8	D	LDT table descriptor
TSDdescKStackSS	+fc8	8	D	SS descriptor
TSDdescFPEM	+fd0	8	D	reserved descriptor slot
TSDdescTIB	+fd8	8	D	FS mapping to TIB
TSDulExitCode	+fe0	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+fe4	4	D	error id from page fault
TSDPFErr	+fe8	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+fec	4	D	
TSDlDbgRangeEnd	+ff0	4	D	
TSDlDbgLastAddr	+ff4	4	D	

Field Name	Offset	Length	Type	Description
TSDpPCB	+ff8	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+ffc	4	D	Pointer to data buffer

3.5.4 Per-Task Data Area for OS/2 Warp V3.0 ALLSTRICT Kernel

For **PTDA** formats for other versions of OS/2 see:

3.5.4.1, “Per-Task Data Area for OS/2 Warp V3.0 RETAIL Kernel” on page 131

3.5.4.2, “Per-Task Data Area for OS/2 V2.11 ALLSTRICT Kernel” on page 136

3.5.4.3, “Per-Task Data Area for OS/2 V2.11 RETAIL Kernel” on page 141

Pointers

TCBpPTDA points to the PTDA associated with a TCB

CurrTSD points to the current TSD.

pPTDASelf points to the current PTDA.

Locations

System Arena.

VM Owner

ptda (0xffcb)

Format

Field Name	Offset	Length	Type	Description
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec′ed)
pPTDANextSibling	+10	4	D	Next sibling′s PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling′s PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	Some TCB before first Tid ′hole′
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header

Field Name	**Offset**	**Length**	**Type**	**Description**
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	Handle to process′s envt seg
ptda_pBeginLIBPATH	+a8	4	D	
ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	1E0	S	
ptda_pPVDBPrc	+290	4	D	
ptda_pSGSList	+294	4	D	
ptda_pexllist	+298	4	D	Flat pointer to exit list data
ptda_cdllterm	+29c	4	D	
WFP_Start	+2a0	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+2a2	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+2a4	2	W	End of Path component of string.
Curr_Dir_End	+2a6	2	W	
CDS_Handle	+2a8	34	W	*REDIR*
OEMPtr	+2dc	2	W	
LIS_Fgnd	+2de	1	B	
FgndOnly	+2df	1	B	Foreground only flag
ptda_pTCBCritSec	+2e0	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+2e4	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+2e8	2	W	Critical Section Count
CurrentPDB	+2ea	2	W	Currently active PDB (V86 segment)
DTAddr	+2ec	4	D	User′s I/O transfer address *REDIR*
seltss	+2f0	2	W	
VolID	+2f2	1	B	!0 if vol ID found in dir search
NoSetDir	+2f3	1	B	If TRUE, do not set directory
SpaceFlag	+2f4	1	B	Embedded spaces allowed in FCB
VerFlg	+2f5	1	B	Initialize with verify off
LCurDrv	+2f6	1	B	Logical current drive - Default A:
PCurDrv	+2f7	1	B	Physical drive after assign mapping
Creating	+2f8	1	B	
DelAll	+2f9	1	B	
FoundDel	+2fa	1	B	

Field Name	**Offset**	**Length**	**Type**	**Description**
Found_dev	+2fb	1	B	True => search found a device 3.10
fSplice	+2fc	1	B	True => do a splice in transpath 3.10
ClusFac	+2fd	1	B	Sectors/cluster used in dir search
cMeta	+2fe	1	B	Components found 3.10
PathNameType	+2ff	1	B	
DevPt	+300	4	D	Address of device found by DevName *REDIR*
DirSec	+304	4	D	
DirStart	+308	2	W	
NxtClusNum	+30a	2	W	
EntFree	+30c	2	W	
EntLast	+30e	2	W	
LastEnt	+310	2	W	
ProcFlag	+312	2	W	If == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+314	4	D	Pending action bits
ptda_ulExitCode	+318	4	D	Exit code of last task
ptda_ulExitType	+31c	4	D	Type of exit
ptda_ulExitTID	+320	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+324	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+328	2	W	SS relative pointer to a curdir struct
CDSsize	+32a	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+32c	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+32e	2	W	Selector of Profile Control Block
ptda_pPCB	+330	4	D	Pointer to Profile Control Block
JFN_Max	+334	2	W	Highest JFN used so far
NextSrchH	+336	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+338	4	D	Handle and Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+33c	2	W	

Field Name	**Offset**	**Length**	**Type**	**Description**
FNotifyLocal_Count	+33e	2	W	
Sig_ignf	+340	2	W	Bit vector of ignored signals
Sig_hndf	+342	2	W	Bit vector of handled signals
Sig_errf	+344	2	W	Bit vector of error generating signals
Sig_attempted	+346	2	W	Bit vector of signals we′ve tried to handle with 32-bit exceptions
Sig_arg	+348	10	W	Byte vector of signal arguments
Sig_termtid	+358	2	W	′Terminator′ TID for APTERM.
HoldSigCnt	+35a	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+35c	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+35e	28	W	Default handle table
JFN_Flags	+386	14	B	Default JFN flags table
ptda_rasflag	+39a	2	W	RAS trace indicator
SysSemPTDATbl	+39c	100	S	
SavedHardErr	+49c	4	D	
ptda_ptdasem	+4a0	C	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4ac	C	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4b8	6	W	Current IDT limit/base
Csid	+4be	2	W	Command Subtree ID
Behav_bit	+4c0	2	W	Program behavior bits
MSW	+4c2	2	W	CPU matching status word
ptda_rsrclist	+4c4	4	D	Far pointer to local resource list
ptda_pldrdldHead	+4c8	4	D	Loader demand load data list
pPrSemTbl	+4cc	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+4d0	4	D	Size of pPrSemTbl in dwords
ulPrTotUsed	+4d4	4	D	Number of entries in pPrSemTbl
ulPrNextFree	+4d8	4	D	Next free slot in pPrSemTbl
hksPrTbl	+4dc	4	D	Kernel semaphore handle for private semaphore table
pShSemBmp	+4e0	4	D	Pointer to private bitmap for the shared semaphore table
ulShBmpSize	+4e4	4	D	Size of pShSemBmp in bits

Field Name	Offset	Length	Type	Description
hksShBmp	+4e8	4	D	Kernel semaphore handle for private semaphore table
ulMtxOwned	+4ec	4	D	Number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+4f0	2	W	Number of share/lock viol retries
RetryCount	+4f2	2	W	Num of share/lock retries to do
RetryLoop	+4f4	2	W	Num of share/lock retry delay loops ceb 75871
ptda_pSrchBuf	+4f6	2	W	Internal search buffer
ptda_pad1	+4f8	2	W	
ptda_pOpenBuf	+4fa	2	W	
ptda_TLMA	+4fc	4	D	In use flag and dword copy count
ptda_TLMABM	+500	4	B	Thread local memory
ptda_TLMASizeMap	+504	20	B	Thread local memory
Cons_Loc	+524	A	S	
SysCallSfcn	+52e	1	B	Value of AL on system entry
SysCall	+52f	1	B	Last system call processed
KBD_Mode	+530	1	B	Keyboard input mode
ptda_NewFiles	+531	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+532	1	B	Non-zero if I 24 FAILed magically
ptda_direntry	+533	20	S	
CP_Flgs	+553	1	B	Default is no codepage in system.
Sig_vec	+554	20	D	Signal handlers
Exc_vec	+574	1C	D	OSOLETE exception vectors
ptda_timerhead	+590	4	D	
Attrib	+594	2	W	Storage for file attributes *REDIR*
ExtFCB	+596	1	B	Extended FCB
ptda_extsig	+597	1	B	
ptda_lanman_sec	+598	4	D	Used by LANMAN and HPFS for security.
ptda_pad2	+59c	2	W	Alignment
ptda_ppgdata	+59e	2	W	
ptda_child	+5a0	2	W	New child PTDA handle (Child being Exec′ed)
ptda_childalias	+5a2	2	W	
ptda_handle	+5a4	2	W	Handle to this segment

Field Name	**Offset**	**Length**	**Type**	**Description**
ptda_module	+5a6	2	W	Program module handle for process
ptda_ldthandle	+5a8	2	W	
ptda_ldtpgmap	+5aa	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5ac	4	D	
CP_CaseMapTbl	+5b0	4	D	
codepage_tag	+5b4	2	W	The current code page
JFN_Length	+5b6	2	W	Size of JFN table in bytes
JFN_pTable	+5b8	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5bc	4	D	Pointer to JFN flags
Joins	+5c0	1	B	Number of joins
ExtErr_Locus	+5c1	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+5c2	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5c4	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5c5	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5c6	24	S	
ptda_pad3	+5ea	2	W	Alignment
CurrTCB	+5ec	2	W	Pointer to current TCB
CurrTSD	+5ee	2	W	Pointer to current TSD
ThisPTDA	+5f0	2	W	Selector for this ptda
ptda_NPX_em_cs	+5f2	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+5f4	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+5f8	2	W	Alignment b726833
ptda_signature	+5fa	2	B	Must contain ″TD″

Table 74 (Page 1 of 2). ptda_ForcedActions Flag Definitions

Name	**Bit Mask**	**Description**
TK_FF_BUF	0x00000001	Buffer must be released
TK_FF_EXIT	0x00000002	Call TKExit (old FF_DES)
TK_FF_CRITSEC	0x00000004	Enter Per-task critical section
TK_FF_ICE	0x00000008	Freeze thread
TK_FF_NPX	0x00000010	NPX Error
TK_FF_TIB	0x00000020	Update the TIB
TK_FF_TRC	0x00000040	Enter Debug
TK_FF_SIG	0x00000080	Signal pending
TK_FF_CTXH	0x00000100	Pending local context hooks

Table 74 (Page 2 of 2). ptda_ForcedActions Flag Definitions

Name	Bit Mask	Description
TK_FF_STIH	0x00000200	Execute STI hooks
TK_FF_VDMBP	0x00000400	Execute VDM BP hooks
TK_FF_RTRY	0x00000800	Retry V86 system call
TK_FF_PIB	0x00001000	Update the PIB
TK_FF_SCH	0x00002000	Do Scheduler Processing
TK_FF_TFBIT	0x00004000	Validate user eflags TF bit
TK_FF_TIBPRI	0x00008000	Update only the priority fields in TIB 59463

3.5.4.1 Per-Task Data Area for OS/2 Warp V3.0 RETAIL Kernel

Field Name	Offset	Length	Type	Description
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec′ed)
pPTDANextSibling	+10	4	D	Next sibling′s PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling′s PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid ′hole′
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process′s envt seg
ptda_pBeginLIBPATH	+a8	4	D	
ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	1E0	S	
ptda_pPVDBPrc	+290	4	D	
ptda_pSGSList	+294	4	D	
ptda_pexllist	+298	4	D	Flat pointer to exit list data

Field Name	Offset	Length	Type	Description
ptda_cdllterm	+29c	4	D	
WFP_Start	+2a0	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+2a2	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+2a4	2	W	End of Path component of string.
Curr_Dir_End	+2a6	2	W	
CDS_Handle	+2a8	34	W	*REDIR*
OEMPtr	+2dc	2	W	
LIS_Fgnd	+2de	1	B	
FgndOnly	+2df	1	B	foreground only flag
ptda_pTCBCritSec	+2e0	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+2e4	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+2e8	2	W	Critical Section Count
CurrentPDB	+2ea	2	W	Currently active PDB (V86 segment)
DTAddr	+2ec	4	D	User's I/O transfer address *REDIR*
seltss	+2f0	2	W	
VolID	+2f2	1	B	!0 if vol ID found in dir search
NoSetDir	+2f3	1	B	If TRUE, do not set directory
SpaceFlag	+2f4	1	B	Embedded spaces allowed in FCB
VerFlg	+2f5	1	B	Initialize with verify off
LCurDrv	+2f6	1	B	Logical current drive - Default A:
PCurDrv	+2f7	1	B	physical drive after assign mapping
Creating	+2f8	1	B	
DelAll	+2f9	1	B	
FoundDel	+2fa	1	B	
Found_dev	+2fb	1	B	true => search found a device 3.10
fSplice	+2fc	1	B	true => do a splice in transpath 3.10
ClusFac	+2fd	1	B	sectors/cluster used in dir search
cMeta	+2fe	1	B	components found 3.10
PathNameType	+2ff	1	B	
DevPt	+300	4	D	Address of device found by DevName *REDIR*
DirSec	+304	4	D	

Field Name	**Offset**	**Length**	**Type**	**Description**
DirStart	+308	2	W	
NxtClusNum	+30a	2	W	
EntFree	+30c	2	W	
EntLast	+30e	2	W	
LastEnt	+310	2	W	
ProcFlag	+312	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+314	4	D	pending action bits
ptda_ulExitCode	+318	4	D	Exit code of last task
ptda_ulExitType	+31c	4	D	Type of exit
ptda_ulExitTID	+320	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+324	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+328	2	W	SS relative pointer to a curdir struct
CDSsize	+32a	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+32c	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+32e	2	W	Selector of Profile Control Block
ptda_pPCB	+330	4	D	Pointer to Profile Control Block
JFN_Max	+334	2	W	highest JFN used so far
NextSrchH	+336	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+338	4	D	Handle and Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+33c	2	W	
FNotifyLocal_Count	+33e	2	W	
Sig_ignf	+340	2	W	bit vector of ignored signals
Sig_hndf	+342	2	W	bit vector of handled signals
Sig_errf	+344	2	W	bit vector of error generating signals
Sig_attempted	+346	2	W	bit vector of signals we′ve tried to handle with 32-bit exceptions
Sig_arg	+348	10	W	byte vector of signal arguments

Field Name	Offset	Length	Type	Description
Sig_termtid	+358	2	W	′Terminator′ TID for APTERM.
HoldSigCnt	+35a	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+35c	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+35e	28	W	default handle table
JFN_Flags	+386	14	B	default JFN flags table
ptda_rasflag	+39a	2	W	RAS trace indicator
SysSemPTDATbl	+39c	100	S	
SavedHardErr	+49c	4	D	
ptda_ptdasem	+4a0	8	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4a8	8	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4b0	6	W	current IDT limit/base
Csid	+4b6	2	W	Command Subtree ID
Behav_bit	+4b8	2	W	program behavior bits
MSW	+4ba	2	W	CPU matching status word
ptda_rsrclist	+4bc	4	D	far pointer to local resource list
ptda_pldrdldHead	+4c0	4	D	loader demand load data list
pPrSemTbl	+4c4	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+4c8	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+4cc	4	D	number of entries in pPrSemTbl
ulPrNextFree	+4d0	4	D	next free slot in pPrSemTbl
hksPrTbl	+4d4	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+4d8	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+4dc	4	D	size of pShSemBmp in bits
hksShBmp	+4e0	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+4e4	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+4e8	2	W	number of share/lock viol retries
RetryCount	+4ea	2	W	num of share/lock retries to do

Field Name	**Offset**	**Length**	**Type**	**Description**
RetryLoop	+4ec	2	W	num of share/lock retry delay loops ceb 75871
ptda_pSrchBuf	+4ee	2	W	internal search buffer
ptda_pad1	+4f0	2	W	
ptda_pOpenBuf	+4f2	2	W	
ptda_TLMA	+4f4	4	D	in use flag and dword copy count
ptda_TLMABM	+4f8	4	B	thread local memory
ptda_TLMASizeMap	+4fc	20	B	thread local memory
Cons_Loc	+51c	A	S	
SysCallSfcn	+526	1	B	Value of AL on system entry
SysCall	+527	1	B	Last system call processed
KBD_Mode	+528	1	B	Keyboard input mode
ptda_NewFiles	+529	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+52a	1	B	Non-zero if I 24 FAILed magically
ptda_direntry	+52b	20	S	
CP_Flgs	+54b	1	B	Default is no codepage in system.
Sig_vec	+54c	20	D	signal handlers
Exc_vec	+56c	1C	D	OSOLETE exception vectors
ptda_timerhead	+588	4	D	
Attrib	+58c	2	W	storage for file attributes *REDIR*
ExtFCB	+58e	1	B	Extended FCB
ptda_extsig	+58f	1	B	
ptda_lanman_sec	+590	4	D	Used by LANMAN and HPFS for security.
ptda_pad2	+594	2	W	alignment
ptda_ppgdata	+596	2	W	
ptda_child	+598	2	W	New child PTDA handle (Child being Exec′ed)
ptda_childalias	+59a	2	W	
ptda_handle	+59c	2	W	handle to this segment
ptda_module	+59e	2	W	program module handle for process
ptda_ldthandle	+5a0	2	W	
ptda_ldtpgmap	+5a2	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5a4	4	D	
CP_CaseMapTbl	+5a8	4	D	
codepage_tag	+5ac	2	W	the current code page

Field Name	Offset	Length	Type	Description
JFN_Length	+5ae	2	W	Size of JFN table in bytes
JFN_pTable	+5b0	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5b4	4	D	pointer to JFN flags
Joins	+5b8	1	B	number of joins
ExtErr_Locus	+5b9	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+5ba	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5bc	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5bd	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5be	24	S	
ptda_pad3	+5e2	2	W	alignment
CurrTCB	+5e4	2	W	pointer to current TCB
CurrTSD	+5e6	2	W	pointer to current TSD
ThisPTDA	+5e8	2	W	Selector for this ptda
ptda_NPX_em_cs	+5ea	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+5ec	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+5f0	2	W	alignment b726833
ptda_signature	+5f2	2	B	must contain ″TD″

3.5.4.2 Per-Task Data Area for OS/2 V2.11 ALLSTRICT Kernel

Field Name	Offset	Length	Type	Description
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec′ed)
pPTDANextSibling	+10	4	D	Next sibling′s PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling′s PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid ′hole′
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays

Field Name	Offset	Length	Type	Description
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process′s envt seg
ptda_pgpc	+a8	400	S	
ptda_pmemstatcur	+4a8	4	D	
ptda_memstat	+4ac	3C	S	
ptda_pPVDBPrc	+4e8	4	D	
ptda_pSGSList	+4ec	4	D	
ptda_pexllist	+4f0	4	D	Flat pointer to exit list data
ptda_cdllterm	+4f4	4	D	
WFP_Start	+4f8	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+4fa	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+4fc	2	W	End of Path component of string.
Curr_Dir_End	+4fe	2	W	
CDS_Handle	+500	34	W	*REDIR*
OEMPtr	+534	2	W	
LIS_Fgnd	+536	1	B	
FgndOnly	+537	1	B	foreground only flag
ptda_pTCBCritSec	+538	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+53c	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+540	2	W	Critical Section Count
CurrentPDB	+542	2	W	Currently active PDB (V86 segment)
DTAddr	+544	4	D	User′s I/O transfer address *REDIR*
seltss	+548	2	W	
VolID	+54a	1	B	!0 if vol ID found in dir search
NoSetDir	+54b	1	B	If TRUE, do not set directory
SpaceFlag	+54c	1	B	Embedded spaces allowed in FCB
VerFlg	+54d	1	B	Initialize with verify off
LCurDrv	+54e	1	B	Logical current drive - Default A:
PCurDrv	+54f	1	B	physical drive after assign mapping
Creating	+550	1	B	

Field Name	Offset	Length	Type	Description
DelAll	+551	1	B	
FoundDel	+552	1	B	
Found_dev	+553	1	B	true => search found a device 3.10
fSplice	+554	1	B	true => do a splice in transpath 3.10
ClusFac	+555	1	B	sectors/cluster used in dir search
cMeta	+556	1	B	components found 3.10
PathNameType	+557	1	B	
DevPt	+558	4	D	Address of device found by DevName *REDIR*
DirSec	+55c	4	D	
DirStart	+560	2	W	
NxtClusNum	+562	2	W	
EntFree	+564	2	W	
EntLast	+566	2	W	
LastEnt	+568	2	W	
ProcFlag	+56a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+56c	4	D	pending action bits
ptda_ulExitCode	+570	4	D	Exit code of last task
ptda_ulExitType	+574	4	D	Type of exit
ptda_ulExitTID	+578	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+57c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+580	2	W	SS relative pointer to a curdir struct
CDSsize	+582	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+584	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+586	2	W	Selector of Profile Control Block
ptda_pPCB	+588	4	D	Pointer to Profile Control Block
JFN_Max	+58c	2	W	highest JFN used so far
NextSrchH	+58e	2	W	Next value to use for search handle First value used will be 2.

Field Name	**Offset**	**Length**	**Type**	**Description**
SrchRmp	+590	4	D	Handle and Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+594	2	W	
FNotifyLocal_Count	+596	2	W	
Sig_ignf	+598	2	W	bit vector of ignored signals
Sig_hndf	+59a	2	W	bit vector of handled signals
Sig_errf	+59c	2	W	bit vector of error generating signals
Sig_attempted	+59e	2	W	bit vector of signals we′ve tried to handle with 32-bit exceptions
Sig_arg	+5a0	10	W	byte vector of signal arguments
Sig_termtid	+5b0	2	W	′Terminator′ TID for APTERM.
HoldSigCnt	+5b2	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+5b4	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+5b6	28	W	default handle table
JFN_Flags	+5de	14	B	default JFN flags table
ptda_rasflag	+5f2	2	W	RAS trace indicator
SysSemPTDATbl	+5f4	100	S	
SavedHardErr	+6f4	4	D	
ptda_ptdasem	+6f8	C	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+704	C	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+710	6	W	current IDT limit/base
Csid	+716	2	W	Command Subtree ID
Behav_bit	+718	2	W	program behavior bits
MSW	+71a	2	W	CPU matching status word
ptda_rsrclist	+71c	4	D	far pointer to local resource list
ptda_pldrdldHead	+720	4	D	loader demand load data list
pPrSemTbl	+724	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+728	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+72c	4	D	number of entries in pPrSemTbl
ulPrNextFree	+730	4	D	next free slot in pPrSemTbl

Field Name	Offset	Length	Type	Description
hksPrTbl	+734	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+738	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+73c	4	D	size of pShSemBmp in bits
hksShBmp	+740	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+744	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+748	2	W	number of share/lock viol retries
RetryCount	+74a	2	W	num of share/lock retries to do
ptda_pad1	+74c	2	W	alignment
ptda_pSrchBuf	+74e	2	W	internal search buffer
ptda_LibiError	+750	2	W	reuse same field to hold library init errors
ptda_pOpenBuf	+752	2	W	
Cons_Loc	+754	A	S	
SysCallSfcn	+75e	1	B	Value of AL on system entry
SysCall	+75f	1	B	Last system call processed
KBD_Mode	+760	1	B	Keyboard input mode
ptda_NewFiles	+761	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+762	1	B	Non-zero if I 24 FAILed magically
ptda_direntry	+763	20	S	
CP_Flgs	+783	1	B	Default is no codepage in system.
Sig_vec	+784	20	D	signal handlers
Exc_vec	+7a4	1C	D	OSOLETE exception vectors
ptda_timerhead	+7c0	4	D	
Attrib	+7c4	2	W	storage for file attributes *REDIR*
ExtFCB	+7c6	1	B	Extended FCB
ptda_extsig	+7c7	1	B	
ptda_lanman_sec	+7c8	4	D	Used by LANMAN and HPFS for security.
ptda_pad2	+7cc	2	W	alignment
ptda_ppgdata	+7ce	2	W	
ptda_child	+7d0	2	W	New child PTDA handle (Child being Exec′ed)

Field Name	Offset	Length	Type	Description
ptda_childalias	+7d2	2	W	
ptda_handle	+7d4	2	W	handle to this segment
ptda_module	+7d6	2	W	program module handle for process
ptda_ldthandle	+7d8	2	W	
ptda_ldtpgmap	+7da	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+7dc	4	D	
CP_CaseMapTbl	+7e0	4	D	
codepage_tag	+7e4	2	W	the current code page
JFN_Length	+7e6	2	W	Size of JFN table in bytes
JFN_pTable	+7e8	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+7ec	4	D	pointer to JFN flags
Joins	+7f0	1	B	number of joins
ExtErr_Locus	+7f1	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+7f2	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+7f4	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+7f5	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+7f6	24	S	
ptda_pad3	+81a	2	W	alignment
CurrTCB	+81c	2	W	pointer to current TCB
CurrTSD	+81e	2	W	pointer to current TSD
ThisPTDA	+820	2	W	Selector for this ptda
ptda_NPX_em_cs	+822	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+824	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+828	2	W	alignment b726833
ptda_signature	+82a	2	B	must contain ″TD″

3.5.4.3 Per-Task Data Area for OS/2 V2.11 RETAIL Kernel

Field Name	Offset	Length	Type	Description
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec′ed)
pPTDANextSibling	+10	4	D	Next sibling′s PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling′s PTDA

Field Name	Offset	Length	Type	Description
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid ′hole′
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process′s envt seg
ptda_pgpc	+a8	400	S	
ptda_pmemstatcur	+4a8	4	D	
ptda_memstat	+4ac	3C	S	
ptda_pPVDBPrc	+4e8	4	D	
ptda_pSGSList	+4ec	4	D	
ptda_pexllist	+4f0	4	D	Flat pointer to exit list data
ptda_cdllterm	+4f4	4	D	
WFP_Start	+4f8	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+4fa	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+4fc	2	W	End of Path component of string.
Curr_Dir_End	+4fe	2	W	
CDS_Handle	+500	34	W	*REDIR*
OEMPtr	+534	2	W	
LIS_Fgnd	+536	1	B	
FgndOnly	+537	1	B	foreground only flag
ptda_pTCBCritSec	+538	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+53c	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+540	2	W	Critical Section Count
CurrentPDB	+542	2	W	Currently active PDB (V86 segment)
DTAddr	+544	4	D	User′s I/O transfer address *REDIR*
seltss	+548	2	W	
VolID	+54a	1	B	!0 if vol ID found in dir search

Field Name	Offset	Length	Type	Description
NoSetDir	+54b	1	B	If TRUE, do not set directory
SpaceFlag	+54c	1	B	Embedded spaces allowed in FCB
VerFlg	+54d	1	B	Initialize with verify off
LCurDrv	+54e	1	B	Logical current drive - Default A:
PCurDrv	+54f	1	B	physical drive after assign mapping
Creating	+550	1	B	
DelAll	+551	1	B	
FoundDel	+552	1	B	
Found_dev	+553	1	B	true => search found a device 3.10
fSplice	+554	1	B	true => do a splice in transpath 3.10
ClusFac	+555	1	B	sectors/cluster used in dir search
cMeta	+556	1	B	components found 3.10
PathNameType	+557	1	B	
DevPt	+558	4	D	Address of device found by DevName *REDIR*
DirSec	+55c	4	D	
DirStart	+560	2	W	
NxtClusNum	+562	2	W	
EntFree	+564	2	W	
EntLast	+566	2	W	
LastEnt	+568	2	W	
ProcFlag	+56a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+56c	4	D	pending action bits
ptda_ulExitCode	+570	4	D	Exit code of last task
ptda_ulExitType	+574	4	D	Type of exit
ptda_ulExitTID	+578	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+57c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+580	2	W	SS relative pointer to a curdir struct
CDSsize	+582	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+584	2	W	Storage for search attrs *REDIR* 3.10

Field Name	Offset	Length	Type	Description
sPCB	+586	2	W	Selector of Profile Control Block
ptda_pPCB	+588	4	D	Pointer to Profile Control Block
JFN_Max	+58c	2	W	highest JFN used so far
NextSrchH	+58e	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+590	4	D	Handle and Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+594	2	W	
FNotifyLocal_Count	+596	2	W	
Sig_ignf	+598	2	W	bit vector of ignored signals
Sig_hndf	+59a	2	W	bit vector of handled signals
Sig_errf	+59c	2	W	bit vector of error generating signals
Sig_attempted	+59e	2	W	bit vector of signals we′ve tried to handle with 32-bit exceptions
Sig_arg	+5a0	10	W	byte vector of signal arguments
Sig_termtid	+5b0	2	W	′Terminator′ TID for APTERM.
HoldSigCnt	+5b2	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+5b4	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+5b6	28	W	default handle table
JFN_Flags	+5de	14	B	default JFN flags table
ptda_rasflag	+5f2	2	W	RAS trace indicator
SysSemPTDATbl	+5f4	100	S	
SavedHardErr	+6f4	4	D	
ptda_ptdasem	+6f8	8	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+700	8	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+708	6	W	current IDT limit/base
Csid	+70e	2	W	Command Subtree ID
Behav_bit	+710	2	W	program behavior bits
MSW	+712	2	W	CPU matching status word
ptda_rsrclist	+714	4	D	far pointer to local resource list

Field Name	Offset	Length	Type	Description
ptda_pldrdldHead	+718	4	D	loader demand load data list
pPrSemTbl	+71c	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+720	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+724	4	D	number of entries in pPrSemTbl
ulPrNextFree	+728	4	D	next free slot in pPrSemTbl
hksPrTbl	+72c	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+730	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+734	4	D	size of pShSemBmp in bits
hksShBmp	+738	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+73c	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+740	2	W	number of share/lock viol retries
RetryCount	+742	2	W	num of share/lock retries to do
RetryLoop	+744	2	W	
ptda_pSrchBuf	+746	2	W	internal search buffer
ptda_LibiError	+748	2	W	reuse same field to hold library init errors
ptda_pOpenBuf	+74a	2	W	
Cons_Loc	+74c	A	S	
SysCallSfcn	+756	1	B	Value of AL on system entry
SysCall	+757	1	B	Last system call processed
KBD_Mode	+758	1	B	Keyboard input mode
ptda_NewFiles	+759	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+75a	1	B	Non-zero if I 24 FAILed magically
ptda_direntry	+75b	20	S	
CP_Flgs	+77b	1	B	Default is no codepage in system.
Sig_vec	+77c	20	D	signal handlers
Exc_vec	+79c	1C	D	OSOLETE exception vectors
ptda_timerhead	+7b8	4	D	
Attrib	+7bc	2	W	storage for file attributes *REDIR*

Field Name	Offset	Length	Type	Description
ExtFCB	+7be	1	B	Extended FCB
ptda_extsig	+7bf	1	B	
ptda_lanman_sec	+7c0	4	D	Used by LANMAN and HPFS for security.
ptda_pad2	+7c4	2	W	alignment
ptda_ppgdata	+7c6	2	W	
ptda_child	+7c8	2	W	New child PTDA handle (Child being Exec′ed)
ptda_childalias	+7ca	2	W	
ptda_handle	+7cc	2	W	handle to this segment
ptda_module	+7ce	2	W	program module handle for process
ptda_ldthandle	+7d0	2	W	
ptda_ldtpgmap	+7d2	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+7d4	4	D	
CP_CaseMapTbl	+7d8	4	D	
codepage_tag	+7dc	2	W	the current code page
JFN_Length	+7de	2	W	Size of JFN table in bytes
JFN_pTable	+7e0	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+7e4	4	D	pointer to JFN flags
Joins	+7e8	1	B	number of joins
ExtErr_Locus	+7e9	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+7ea	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+7ec	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+7ed	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+7ee	24	S	
ptda_pad3	+812	2	W	alignment
CurrTCB	+814	2	W	pointer to current TCB
CurrTSD	+816	2	W	pointer to current TSD
ThisPTDA	+818	2	W	Selector for this ptda
ptda_NPX_em_cs	+81a	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+81c	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+820	2	W	alignment b726833
ptda_signature	+822	2	B	must contain ″TD″

3.5.5 Local Information Segement

Pointers

SAS field **SAS_info_local** points to the current LISEG.

Locations

dfff:0 is the address of the copy of the LISEG for the current thread and process.

The LISEG for each process is imbedded in the PTDA at **ptda_infoseg**.

VM Owner

infoseg (0xff75)

Format

Table 75 (Page 1 of 2). InfoSegLDT

Field Name	Offset	Length	Type	Description
LIS_CurProcID	+0	2	W	Current process ID
LIS_ParProcID	+2	2	W	Process ID of parent
LIS_CurThrdPri	+4	2	W	Current thread priority
LIS_CurThrdID	+6	2	W	Current thread ID
LIS_CurScrnGrp	+8	2	W	Screengroup
LIS_ProcStatus	+a	1	B	Process status bits
LIS_fillbyte1	+b	1	B	filler byte
LIS_Fgnd	+c	2	W	Current process is in foreground
LIS_ProcType	+e	1	B	Current process type
LIS_fillbyte2	+f	1	B	filler byte
LIS_AX	+10	2	W	@@V1 Environment selector
LIS_BX	+12	2	W	@@V1 Offset of command line start
LIS_CX	+14	2	W	@@V1 Length of Data Segment
LIS_DX	+16	2	W	@@V1 STACKSIZE from the .EXE file
LIS_SI	+18	2	W	@@V1 HEAPSIZE from the .EXE file
LIS_DI	+1a	2	W	@@V1 Module handle of the application
LIS_DS	+1c	2	W	@@V1 Data Segment Handle of application
LIS_PackSel	+1e	2	W	First tiled selector in this EXE
LIS_PackShrSel	+20	2	W	First selector above shared arena

Table 75 (Page 2 of 2). InfoSegLDT

Field Name	**Offset**	**Length**	**Type**	**Description**
LIS_PackPckSel	+22	2	W	First selector above packed arena

Table 76. LIS_ProcStatus Flag Definitions

Name	**Bit Mask**	**Description**
PS_XITLST	0x01	Doing ExitList Processing
PS_XITTH1	0x02	Exiting thread 1
PS_XITALL	0x04	The whole process is exiting
PS_SYNCPARENT	0x10	Parent cares about termination
PS_WAITPARENT	0x20	Parent did an exec-and-wait
PS_DYING	0x40	Process is dying
PS_EMBRYO	0x80	Process in embryonic state

Table 77. LIS_ProcType Flag Definitions

Name	**Value**	**Description**
LIS_PT_FULLSCRN	0	Full screen app.
LIS_PT_REALMODE	1	Real mode process
PT_VDM	1	VDM
LIS_PT_VIOWIN	2	VIO windowable app.
LIS_PT_PRESMGR	3	Presentation Manager app.
LIS_PT_DETACHED	4	Detached app.

3.5.6 Global Information Segement

Pointers

SAS field **SAS_info_global** points to the current GISEG.

Locations

dff4:0 is the address of the copy of the GISEG for the current thread and process.

VM Owner

infoseg (0xff75) - shared arena copy

os2krnl (0xffaa) - system arena copy

Format

Table 78 (Page 1 of 2). InfoSegGDT

Field Name	Offset	Length	Type	Description
SIS_BigTime	+0	4	D	Time from 1-1-1970 in seconds
SIS_MsCount	+4	4	D	Freerunning milliseconds counter
SIS_HrsTime	+8	1	B	Hours
SIS_MinTime	+9	1	B	Minutes
SIS_SecTime	+a	1	B	Seconds
SIS_HunTime	+b	1	B	Hundredths of seconds
SIS_TimeZone	+c	2	W	Timezone in min from GMT (Set to EST)
SIS_ClkIntrvl	+e	2	W	Timer interval (units=0.0001 secs)
SIS_DayDate	+10	1	B	Day-of-month (1-31)
SIS_MonDate	+11	1	B	Month (1-12)
SIS_YrsDate	+12	2	W	Year (>= 1980)
SIS_DOWDate	+14	1	B	Day-of-week (1-1-80 = Tues = 3)
SIS_VerMajor	+15	1	B	Major version number
SIS_VerMinor	+16	1	B	Minor version number
SIS_RevLettr	+17	1	B	Revision letter
SIS_CurScrnGrp	+18	1	B	Fgnd screen group #
SIS_MaxScrnGrp	+19	1	B	Maximum number of screen groups
SIS_HugeShfCnt	+1a	1	B	Shift count for huge segments
SIS_ProtMdOnly	+1b	1	B	Protect-mode-only indicator
SIS_FgndPID	+1c	2	W	Foreground process ID
SIS_Dynamic	+1e	1	B	Dynamic variation flag (1=enabled)
SIS_MaxWait	+1f	1	B	Maxwait (seconds)
SIS_MinSlice	+20	2	W	Minimum timeslice (milliseconds)
SIS_MaxSlice	+22	2	W	Maximum timeslice (milliseconds)
SIS_BootDrv	+24	2	W	Drive from which system was booted
SIS_mec_table	+26	20	B	Table of RAS Major Event Codes (MECs)
SIS_MaxVioWinSG	+46	1	B	Max. no. of VIO windowable SG′s
SIS_MaxPresMgrSG	+47	1	B	Max. no. of Presentation Manager SG′s
SIS_SysLog	+48	2	W	Error Logging Status

Table 78 (Page 2 of 2). InfoSegGDT				
Field Name	**Offset**	**Length**	**Type**	**Description**
SIS_MMIOBase	+4a	2	W	Memory mapped I/O selector
SIS_MMIOAddr	+4c	4	D	Memory mapped I/O address
SIS_MaxVDMs	+50	1	B	Max. no. of Virtual DOS machines
SIS_Reserved	+51	1	B	

Table 79. SIS_SysLog Flag Definitions

Name	**Bit Mask**	**Description**
LF_LOGENABLE	0x0001	Logging enabled
LF_LOGAVAILABLE	0x0002	Logging available

3.5.7 Process Information Block

Pointers

PTDA field **ptda_avatib** points to the PIB for the related process.

PIB field **pib_pchenv** points to the process′ environment strings.

Locations

Allocated in the process′ private arena.

VM Owner

PIB owner id: **tktib (0xff3f)** (also used for TIB ownership).

Environment Owner ID: **tkenv (0xff3e)**.

Format

Table 80. PIB Process Information Block

Field Name	**Offset**	**Length**	**Type**	**Description**
pib_ulpid	+0	4	D	Process ID
pib_ulppid	+4	4	D	Parent process I.D.
pib_hmte	+8	4	D	Program (.EXE) module handle
pib_pchcmd	+c	2	W	Command line pointer
pib_pchenv	+10	4	D	Environment pointer
pib_flstatus	+14	4	D	Process′ status bits
pib_ultype	+18	4	D	Process′ type code

Table 81. pib_flstatus Flag Definitions

Name	**Bit Mask**	**Description**
PS_XITLST	0x01	Doing ExitList Processing
PS_XITTH1	0x02	Exiting thread 1
PS_XITALL	0x04	The whole process is exiting
PS_SYNCPARENT	0x10	Parent cares about termination
PS_WAITPARENT	0x20	Parent did an exec-and-wait
PS_DYING	0x40	Process is dying
PS_EMBRYO	0x80	Process in embryonic state

Table 82. pib_ultype Flag Definitions

Name	**Value**	**Description**
LIS_PT_FULLSCRN	0	Full screen app.
LIS_PT_REALMODE	1	Real mode process
PT_VDM	1	VDM
LIS_PT_VIOWIN	2	VIO windowable app.
LIS_PT_PRESMGR	3	Presentation Manager app.
LIS_PT_DETACHED	4	Detached app.

3.5.8 Thread Information Block

Pointers

TCB field **TCBptib** points to the TIB for the related thread.

TIB field **tib_ptib2** points to the associated TIB2.

GDT Selector 150b maps the TIB and is the default value for the FS register.

Locations

Allocated in the process′ private arena.

VM Owner

tktib (0xff3f) (also used for PIB ownership).

Format

TIB Thread Information Block system independent section.

Field Name	Offset	Length	Type	Description
tib_pexchain	+0	4	D	Head of exception handler chain
tib_pstack	+4	4	D	Pointer to base of stack
tib_pstacklimit	+8	4	D	Pointer to end of stack
tib_ptib2	+c	2	W	Pointer to system specific TIB
tib_version	+10	4	D	Version number for this TIB structure
tib_ordinal	+14	4	D	Thread Ordinal Number

TIB2 Thread Information Block system dependent section.

Field Name	Offset	Length	Type	Description
tib2_ultid	+0	4	D	Thread ID
tib2_ulpri	+4	4	D	Thread priority
tib2_version	+8	4	D	Version number for this structure
tib2_usMCCount	+c	2	W	Must Complete count
tib2_fMCForceFlag	+e	2	W	Must Complete force flag

3.5.9 System Stack Frames Client Register Information

Pointers
TCB field **TCB_pcriFrameType** points to the CRI.

Locations
_criISF locates the Interrupt Stack Frame CRI.

_criTSF locates the Trap Stack Frame CRI.

_criVSF locates the VDM Stack Frame CRI.

_criSEF locates the System Entry Stack Frame CRI.

_criPASCALSEF locates the PASCAL System Entry Stack Frame CRI.

_criSSF locates the SCI Stack Frame CRI.

_criDHF locates the Device Help Stack Frame CRI.

fpoldstack contains a 32-bit far pointer to the ISF built by the Interrupt Router at interrupt time.

VM Owner
os2krnl (0xffaa)

Format

Table 83. CRI Client Register Information

Field Name	**Offset**	**Length**	**Type**	**Description**
cri_ulSize	+0	4	D	size of stack frame
cri_eax	+4	4	S	eax rip
cri_ebx	+8	4	S	ebx rip
cri_ecx	+c	4	S	ecx rip
cri_edx	+10	4	S	edx rip
cri_ebp	+14	4	S	ebp rip
cri_esi	+18	4	S	esi rip
cri_edi	+1c	4	S	edi rip
cri_ds	+20	4	S	ds rip
cri_es	+24	4	S	es rip
cri_fs	+28	4	S	fs rip
cri_gs	+2c	4	S	gs rip
cri_cs	+30	4	S	cs rip
cri_eip	+34	4	S	eip rip
cri_eflag	+38	4	S	eflag rip
cri_ss	+3c	4	S	ss rip
cri_esp	+40	4	S	esp rip
cri_cbargs	+44	4	S	cbargs rip
cri_trapnum	+48	4	S	trapnum rip
cri_errcode	+4c	4	S	errcode rip
cri_pfnRebuild	+50	4	D	
cri_pfpfnKernelExit	+54	4	D	

Table 84. RIP Register Information Packet

Field Name	**Offset**	**Length**	**Type**	**Description**
rip_flags	+0	2	W	Flags
rip_offset	+2	4	W	Offset of register into stack frame

Table 85 (Page 1 of 2). rip_flags Flag Definitions

Name	**Bit Mask**	**Description**
KM_RIP_INVALID	0x0001	invalid register
KM_RIP_INVALID_SET	0x0002	invalid register to set
KM_RIP_WORD	0x0004	word register
KM_RIP_TSD_RELATIVE	0x0008	rip_offset relative to TSD beginning

Table 85 (Page 2 of 2). rip_flags Flag Definitions

Name	**Bit Mask**	**Description**
KM_RIP_C32	0x0010	32-bit C style call

ISF Interrupt Manager Stack Frame

This is what the stack frame looks like when the system is entered through the interrupt manager during a hardware interrupt. For a hardware interrupt in a VDM context, the stack frame always needs to be a ″VSF″ type so the stack frame base is adjusted by ISF_VSF_START. The points the stack frame base to ″isf_edi″ in the regular interrupt frame. The interrupt stack frame has also been padded (ISF_STACK_PAD) between the general registers (EDI to EAX) and the hardware pushed registers (EIP to SS) with a dummy trap number and error code to look like the VSF stack frame.

Table 86. ISF Interrupt Manager Stack Frame

Field Name	**Offset**	**Length**	**Type**	**Description**
isf_CurrIntLevel	+0	4	D	
isf_gs	+4	2	W	
isf_padgs	+6	2	W	
isf_fs	+8	2	W	
isf_padfs	+a	2	W	
isf_es	+c	2	W	
isf_pades	+e	2	W	
isf_ds	+10	2	W	
isf_padds	+12	2	W	
isf_edi	+14	4	D	start of VDM stack frame
isf_esi	+18	4	D	
isf_ebp	+1c	4	D	
isf_padesp	+20	4	D	
isf_ebx	+24	4	D	
isf_edx	+28	4	D	
isf_ecx	+2c	4	D	
isf_eax	+30	4	D	
isf_pad	+34	8	B	
isf_eip	+3c	4	D	
isf_cs	+40	2	W	
isf_padcs	+42	2	W	
isf_eflag	+44	4	D	
isf_esp	+48	4	D	
isf_ss	+4c	2	W	
isf_padss	+4e	2	W	

TSF Trap or Exception Stack Frame

This is what the stack frame looks like when the system is entered through a 386 exception (from protected mode).

Table 87. TSF Trap or Exception Stack Frame

Field Name	Offset	Length	Type	Description
tsf_edi	+0	4	D	
tsf_esi	+4	4	D	
tsf_ebp	+8	4	D	
tsf_padesp	+c	4	D	
tsf_ebx	+10	4	D	
tsf_edx	+14	4	D	
tsf_ecx	+18	4	D	
tsf_eax	+1c	4	D	
tsf_gs	+20	2	W	
tsf_padgs	+22	2	W	
tsf_fs	+24	2	W	
tsf_padfs	+26	2	W	
tsf_es	+28	2	W	
tsf_pades	+2a	2	W	
tsf_ds	+2c	2	W	
tsf_padds	+2e	2	W	
tsf_trapnum	+30	4	D	
tsf_errcode	+34	4	D	
tsf_eip	+38	4	D	
tsf_cs	+3c	2	W	
tsf_padcs	+3e	2	W	
tsf_eflag	+40	4	D	
tsf_esp	+44	4	D	
tsf_ss	+48	2	W	
tsf_padss	+4a	2	W	

KSF Kernel Stack Frame

This is what the stack frame looks like when the system is re-entered from ring 0. This is frame used for handling exception while already in kernel mode.

Table 88. KSF Kernel Stack Frame

Field Name	Offset	Length	Type	Description
ksf_edi	+0	4	D	
ksf_esi	+4	4	D	
ksf_ebp	+8	4	D	
ksf_padesp	+c	4	D	
ksf_ebx	+10	4	D	
ksf_edx	+14	4	D	
ksf_ecx	+18	4	D	
ksf_eax	+1c	4	D	
ksf_gs	+20	2	W	
ksf_padgs	+22	2	W	
ksf_fs	+24	2	W	
ksf_padfs	+26	2	W	
ksf_es	+28	2	W	
ksf_pades	+2a	2	W	
ksf_ds	+2c	2	W	
ksf_padds	+2e	2	W	
ksf_trapnum	+30	4	D	
ksf_errcode	+34	4	D	
ksf_eip	+38	4	D	
ksf_cs	+3c	2	W	
ksf_padcs	+3e	2	W	
ksf_eflag	+40	4	D	

VSF VDM Process Stack Frame

This is what the stack frame looks like when the system is entered from a VDM through an exception, software or hardware interrupt. Most of the 8086 emulation code uses this stack frame directly for performance. For hardware interrupts taken in a VDM (in either V86 mode or protected mode), the interrupt stack frame (see ISF) is adjusted to look like this frame.

The alternate stack frame holds the real or protected mode sensitive registers for the other mode. So when the VDM is in protected mode, the last V86 mode segment registers CS:EIP and SS:ESP can be accessed. Two things happen with the mode switch: 1) the alternate register set is exchanged with the regular set (vsf_eip to vsf_padgs is the exchanged with vsf_alteip to vsf_altpadgs), 2) the TSS′s ESP0 value is changed to the appropriate place in the VSF structure. For V86 mode, ESP0 points to the begining of the segment registers (vsf_gs/vsf_padgs) and for protected mode ESP0 points to the SS register (vsf_ss/vsf_padss). For protected mode entry, the segments registers are stored in vsf_ds to vsf_gs explictly. This makes the V86 mode and protected mode stack frames the same for VDDs and the MVDM kernel code.

Table 89. VSF VDM Process Stack Frame

Field Name	Offset	Length	Type	Description
vsf_edi	+0	4	D	
vsf_esi	+4	4	D	
vsf_ebp	+8	4	D	
vsf_padesp	+c	4	D	
vsf_ebx	+10	4	D	
vsf_edx	+14	4	D	
vsf_ecx	+18	4	D	
vsf_eax	+1c	4	D	
vsf_trapnum	+20	4	D	
vsf_errcode	+24	4	D	
vsf_eip	+28	4	D	
vsf_cs	+2c	2	W	
vsf_padcs	+2e	2	W	
vsf_eflag	+30	4	D	
vsf_esp	+34	4	D	
vsf_ss	+38	2	W	
vsf_padss	+3a	2	W	
vsf_es	+3c	2	W	
vsf_pades	+3e	2	W	
vsf_ds	+40	2	W	
vsf_padds	+42	2	W	
vsf_fs	+44	2	W	
vsf_padfs	+46	2	W	
vsf_gs	+48	2	W	
vsf_padgs	+4a	2	W	
vsf_alteip	+4c	4	D	
vsf_altcs	+50	2	W	
vsf_altpadcs	+52	2	W	
vsf_alteflag	+54	4	D	
vsf_altesp	+58	4	D	
vsf_altss	+5c	2	W	
vsf_altpadss	+5e	2	W	
vsf_altes	+60	2	W	
vsf_altpades	+62	2	W	
vsf_altds	+64	2	W	
vsf_altpadds	+66	2	W	
vsf_altfs	+68	2	W	
vsf_altpadfs	+6a	2	W	
vsf_altgs	+6c	2	W	
vsf_altpadgs	+6e	2	W	

SEF System Entry Stack Frame

This is the frame put on the by the call gate system entry function (KMEnterKmodeCallGate or KMEnterKmodeAPI32).

This frame is used for:

- 32-bit C APIs, with C callable workers (criSEF)
- 16-bit PASCAL APIs, with C callable workers (criPASCALSEF)

Table 90. SEF System Entry Stack Frame

Field Name	**Offset**	**Length**	**Type**	**Description**
sef_edi	+0	4	D	
sef_esi	+4	4	D	
sef_ebp	+8	4	D	
sef_padesp	+c	4	D	
sef_ebx	+10	4	D	
sef_edx	+14	4	D	
sef_ecx	+18	4	D	
sef_eax	+1c	4	D	
sef_gs	+20	2	W	
sef_padgs	+22	2	W	
sef_fs	+24	2	W	
sef_padfs	+26	2	W	
sef_es	+28	2	W	
sef_pades	+2a	2	W	
sef_ds	+2c	2	W	
sef_padds	+2e	2	W	
sef_retaddr	+30	4	D	
sef_cbargs	+34	4	D	
sef_eflag	+38	4	D	
sef_eip	+3c	4	D	
sef_cs	+40	2	W	
sef_padcs	+42	2	W	

SCI System Call Interpreter Call Gate Stack Frame

This is what the stack frame looks like when the system is entered through SCI via call gate using the KMEnterKmodeSCI function.

Table 91. SCI System Call Interpreter Call Gate Stack Frame

Field Name	**Offset**	**Length**	**Type**	**Description**
ssf_edi	+0	4	D	
ssf_esi	+4	4	D	
ssf_ebp	+8	4	D	
ssf_padesp	+c	4	D	
ssf_ebx	+10	4	D	
ssf_edx	+14	4	D	
ssf_ecx	+18	4	D	
ssf_eax	+1c	4	D	
ssf_gs	+20	2	W	
ssf_padgs	+22	2	W	
ssf_fs	+24	2	W	
ssf_padfs	+26	2	W	
ssf_es	+28	2	W	
ssf_pades	+2a	2	W	
ssf_ds	+2c	2	W	
ssf_padds	+2e	2	W	
ssf_thopadr	+30	4	D	
ssf_cbargs	+34	4	D	The Most Significant Bit of cbargs in an SCI stack frame is used to denote that a 16-bit callgate is being used with the SCI mechanism. Used by dynamic APIs.
ssf_sciret	+38	2	W	
ssf_eflag	+3a	4	D	
ssf_eip	+3e	4	D	
ssf_cs	+42	2	W	
ssf_padcs	+42	2	W	

Table 92 (Page 1 of 2). DHF Device Help Stack Frame

Field Name	**Offset**	**Length**	**Type**	**Description**
dhf_edi	+0	4	D	
dhf_esi	+4	4	D	
dhf_ebp	+8	4	D	
dhf_padesp	+c	4	D	

Table 92 (Page 2 of 2). DHF Device Help Stack Frame

Field Name	Offset	Length	Type	Description
dhf_ebx	+10	4	D	
dhf_edx	+14	4	D	
dhf_ecx	+18	4	D	
dhf_eax	+1c	4	D	
dhf_gs	+20	2	W	
dhf_padgs	+22	2	W	
dhf_fs	+24	2	W	
dhf_padfs	+26	2	W	
dhf_es	+28	2	W	
dhf_pades	+2a	2	W	
dhf_ds	+2c	2	W	
dhf_padds	+2e	2	W	
dhf_eflag	+30	4	D	
dhf_scratch	+34	4	D	
dhf_eip	+38	4	D	
dhf_cs	+3c	2	W	
dhf_padcs	+3e	4	D	

TF Hardware Exception Trap Stack Frame

Stack frame for the trap manager before we go into kernel mode.

Table 93. TF Hardware Exception Trap Stack Frame

Field Name	Offset	Length	Type	Description
tf_trapnum	+0	4	D	
tf_errcode	+4	4	D	
tf_eip	+8	4	D	
tf_cs	+c	2	W	
tf_padcs	+e	2	W	
tf_eflags	+10	4	D	
tf_esp	+14	4	D	
tf_ss	+18	2	W	
tf_padss	+1a	2	W	

3.5.10 Exit List Entry Data Structure

Pointers
PTDA field **ptda_pexllist** points to the head of Exit List chain.

Locations
Allocated from the kernel heaps.

VM Owner
tkextlst (0xffc7)

Format

Table 94. EXENT

Field Name	Offset	Length	Type	Description
exl_next	+ 0	4	D	link to next block/order
exl_addr	+ 4	4	D	Exit list routine address
exl_class	+ 8	2	W	order and position 0 thru 0x1FF
exl_type	+ a	2	W	16:16 or 0:32

Table 95. exl_type Values

Name	Value	Description
TK_TYPE16	0	
TK_TYPE32	1	
TK_TYPEDT	2	

3.5.11 Exception Handler Structures

Pointers
TIB field **tib_pexchain** points to the head of the chain of EXCEPTIONREGISTRATIONRECORDs.

The first parameter to the exception handler points to the EXCEPTIONREPORTRECORD.

The second parameter to the exception handler points to the EXCEPTIONREGISTRATIONRECORD.

The third parameter to the exception handler points to the CONTEXTRECORD.

Locations
Allocated in the Process′ Private Arena,.

VM Owner
tktib (0xff3f) (also used for PIB ownership).

Format

Table 96. CONTEXTRECORD Exception Handler Context Record

Field Name	Offset	Length	Type	Description
ContextFlags	+0	4	D	Flags
	+4	6c	S	Floating point section
ctx_env	+4	1c	D	Floating point environment
ctx_stack	+20	50	S	Floating point register stack (8 FPREG structures)
	+70	10	S	Segment Register section
ctx_SegGs	+70	4	D	GS segment register
ctx_SegFs	+74	4	D	FS segment register
ctx_SegEs	+78	4	D	ES segment register
ctx_SegDs	+7c	4	D	DS segment register
	+80	18	S	Integer Register section
ctx_RegEdi	+80	4	D	EDI register
ctx_RegEsi	+84	4	D	ESI register
ctx_RegEax	+88	4	D	EAX register
ctx_RegEbx	+8c	4	D	EBX register
ctx_RegEcx	+90	4	D	ECX register
ctx_RegEdx	+94	4	D	EDX register
	+98	18	S	Control Register section
ctx_RegEbp	+98	4	D	EBP register
ctx_RegEip	+9c	4	D	EIP register
ctx_SegCs	+a0	4	D	CS selector
ctx_EFlags	+a4	4	D	Processor Flags register
ctx_RegEsp	+a8	4	D	ESP register
ctx_SegSs	+ac	4	D	SS segment register

Table 97 (Page 1 of 2). FPREG Floating Point Register Stack Element

Field Name	Offset	Length	Type	Description
losig	+0	4	D	Low significance doubleword
hisig	+4	4	D	High significance doubleword

Table 97 (Page 2 of 2). FPREG Floating Point Register Stack Element

Field Name	Offset	Length	Type	Description
signexp	+ 8	2	W	Exponent

Table 98. ContextFlags Flag Definitions

Name	Bit Mask	Description
CONTEXT_CONTROL	0x00000001L	SS:ESP, CS:EIP, EFLAGS, EBP
CONTEXT_INTEGER	0x00000002L	EAX, EBX, ECX, EDX, ESI, EDI
CONTEXT_SEGMENTS	0x00000004L	DS, ES, FS, GS
CONTEXT_FLOATING_POINT	0x00000008L	Numeric coprocessor state

Table 99. EXCEPTIONREPORTRECORD Exception Handler Report Record

Field Name	Offset	Length	Type	Description
ExceptionNum	+ 0	4	D	Exception number
fHandlerFlags	+ 4	4	D	Exception attributes
NestedExceptionReportRecord	+ 8	4	D	Preceding exception's report record if nested exception
ExceptionAddress	+ c	4	D	Exception address
cParameters	+ 10	4	D	Size of exception specific information
ExceptionInfo	+ 14	10	D	Exception specfic information

Table 100. fHandlerFlags Flag Definitions

Name	Bit Mask	Description
EH_NONCONTINUABLE	0x1	Noncontinuable exception
EH_UNWINDING	0x2	Unwind is in progress
EH_EXIT_UNWIND	0x4	Exit unwind is in progress
EH_STACK_INVALID	0x8	Stack out of limits or unaligned
EH_NESTED_CALL	0x10	Nested exception handler call

Table 101. EXCEPTIONREGISTRATIONRECORD Exception Handler Registration Record

Field Name	**Offset**	**Length**	**Type**	**Description**
prev_structure	+ 0	4	D	Previously registered exception handler
ExceptionHandler	+ 4	4	D	Exception handler entry point address or -1 if end of chain

3.6 Loader Control Block Reference

The following control blocks are described in this section:

3.6.1 Module Table Entry for OS/2 Warp V3.0

For **MTE** formats for other versions of OS/2 see:

3.6.1.1, "Module Table Entry for OS/2 V2.11" on page 167

Pointers

_mte_h points to the head of the chain of MTEs.

_global_h points to head of the chain of library module MTEs.

Locations

Dynamically allocated from the kernel resident heap except for the two MTEs that represent kernel interfaces.

_DosMosMte locates the MTE in OS2KRNL for DOSCALLS.DLL.

_VDDModMte locates the MTE in OS2KRNL for MVDM.DLL.

VM Owner

Dynamically allocated MTEs have owner ID **ldrmte (0xffa6)**

Format

Field Name	Offset	Length	Type	Description
mte_flags2	+0	2	W	Module flags 2
mte_handle	+2	2	W	the handle for this mte
mte_swapmte	+4	4	D	link to swappable mte
mte_link	+8	4	D	link to next mte
mte_flags1	+c	4	D	Module flags 1
mte_impmodcnt	+10	4	D	Num of entries in Imp Mod Name Tbl
mte_sfn	+14	2	W	file system number for open file
mte_usecnt	+16	2	W	.EXE only - use count
mte_modname	+18	8	B	resident module name (zero extended)

Table 102. mte_flags1 Flag Definitions

Name	**Bit Mask**	**Description**
NOAUTODS	0x00000000	No Auto DS exists
SOLO	0x00000001	Auto DS is shared
INSTANCEDS	0x00000002	Auto DS is not shared
INSTLIBINIT	0x00000004	Per-instance Libinit
GINISETUP	0x00000008	Global Init has been setup
NOINTERNFIXUPS	0x00000010	internal fixups in .EXE-.DLL applied
NOEXTERNFIXUPS	0x00000020	external fixups in .EXE-.DLL applied
CLASS_PROGRAM	0x00000040	Program class
CLASS_GLOBAL	0x00000080	Global class
CLASS_SPECIFIC	0x000000C0	Specific class, as against global
CLASS_ALL	0x00000000	nonspecific class - all modules
CLASS_MASK	0x000000C0	
MTEPROCESSED	0x00000100	MTE being loaded
USED	0x00000200	MTE is referenced - see ldrgc.c
DOSLIB	0x00000400	set if DOSCALL1
DOSMOD	0x00000800	set if DOSCALLS
MTE_MEDIAFIXED	0x00001000	File Media permits discarding
LDRINVALID	0x00002000	module not loadable
PROGRAMMOD	0x00000000	program module
DEVDRVMOD	0x00004000	device driver module
LIBRARYMOD	0x00008000	DLL module
VDDMOD	0x00010000	VDD module
MVDMMOD	0x00020000	Set if VDD Helper MTE (MVDM.DLL)
INGRAPH	0x00040000	In Module Graph - see ldrgc.c
GINIDONE	0x00080000	Global Init has finished
MTEADDRALLOCED	0x00100000	Allocate specific or not
FSDMOD	0x00200000	FSD MTE
FSHMOD	0x00400000	FS helper MTE
MTELONGNAMES	0x00800000	Module supports long-names
MTE_MEDIACONTIG	0x01000000	File Media contiguous memory req
MTE_MEDIA16M	0x02000000	File Media requires mem below 16M
MTESWAPONLOAD	0x04000000	make code pages swap on load
MTEPORTHOLE	0x08000000	porthole module
MTEMODPROT	0x10000000	Module has shared memory protected
MTENEWMOD	0x20000000	Newly added module
MTEDLLTERM	0x40000000	Gets instance termination
MTESYMLOADED	0x80000000	Set if debugger symbols loaded

Table 103. mte_flags2 Flag Definitions

Name	Bit Mask	Description
MTEFORMATMASK	0x0003	Module format mask
MTEFORMATR1	0x0000	Module format reserved
MTEFORMATNE	0x0001	Module format NE
MTEFORMATLX	0x0002	Module format LX
MTEFORMATR2	0x0003	Module format reserved
MTESYSTEMDLL	0x0004	DLL exists in system list
MTELOADORATTACH	0x0008	Module under load or attach - for init
MTECIRCLEREF	0x0010	Module circular reference detection
MTEFREEFIXUPS	0x0020	Free system mte's fixup flag d#98488
MTEPRELOADED	0x0040	MTE Preload completed
MTEGETMTEDONE	0x0080	GetMTE already resolved
MTEPACKSEGDONE	0x0100	Segment packed memory allocated
MTE20LIELIST	0x0200	Name present in version20 lie list
MTESYSPROCESSED	0x0400	System DLL already processed
MTEDLLONEXTLST	0x1000	DLL has term routine on exit list #74177 removed - 75809

3.6.1.1 Module Table Entry for OS/2 V2.11

Field Name	Offset	Length	Type	Description
mte_flags2	+0	2	W	Module flags 2
mte_handle	+2	2	W	the handle for this mte
mte_swapmte	+4	4	D	link to swappable mte
mte_modname	+8	4	A	resident module name (zero extended)
mte_link	+c	4	D	link to next mte
mte_flags1	+10	4	D	Module flags 1
mte_impmodcnt	+14	4	D	Num of entries in Imp Mod Name Tbl
mte_sfn	+18	2	W	file system number for open file
mte_usecnt	+1a	2	W	.EXE only - use count

3.6.2 Swappable Module Table Entry for OS/2 Warp V3.0

Pointers

MTE field **mte_swapmte** points to the associated SMTE.

Locations

Dynamically allocated from the kernel swappable heap except for the SMTE associalted with DOSCALLS.DLL.

_DosMosMteSwappable locates the SMTE in OS2KRNL associated with DOSCALLS.DLL.

VM Owner

Dynamically allocated SMTEs have owner ID **ldrmte (0xffa6)**

Format

Field Name	Offset	Length	Type	Description
smte_mpages	+0	4	D	Module # pages
smte_startobj	+4	4	D	Object # for instruction pointer
smte_eip	+8	4	D	Extended instruction pointer
smte_stackobj	+c	4	D	Object # for stack pointer
smte_esp	+10	4	D	Extended stack pointer
smte_pageshift	+14	4	D	
smte_fixupsize	+18	4	D	Fixup section size
smte_objtab	+1c	4	D	Object table offset
smte_objcnt	+20	4	D	Number of objects in module
smte_objmap	+24	4	D	Object page map offset
smte_itermap	+28	4	D	Object iterated data map offset
smte_rsrctab	+2c	4	D	Offset of Resource Table
smte_rsrccnt	+30	4	D	Number of resource entries
smte_restab	+34	4	D	Offset of resident name table
smte_enttab	+38	4	D	Offset of Entry Table
smte_fpagetab	+3c	4	D	Offset of Fixup Page Table
smte_frectab	+40	4	D	Offset of Fixup Record Table
smte_impmod	+44	4	D	Offset of Import Module Name Table
smte_impproc	+48	4	D	Offset of Imp Procedure Name Tab
smte_datapage	+4c	4	D	Offset of Enumerated Data Pages
smte_nrestab	+50	4	D	Offset of Non-resident Names Table
smte_cbnrestab	+54	4	D	Size of Non-resident Name Table
smte_autods	+58	4	D	Object # for automatic data object
smte_debuginfo	+5c	4	D	Offset of the debugging info
smte_debuglen	+60	4	D	The len of the debug info in bytes
smte_heapsize	+64	4	D	use for converted 16-bit modules

Field Name	Offset	Length	Type	Description
smte_path	+68	4	D	full pathname
smte_semcount	+6c	2	W	Count of threads waiting on MTE semaphore. 0 => semaphore is free
smte_semowner	+6e	2	W	Slot number of the owner of MTE semahore
smte_pfilecache	+70	4	D	Pointer to file cache for Dos32CacheModule
smte_stacksize	+74	4	D	Thread 1 Stack size from the exe header
smte_alignshift	+78	2	W	use for converted 16-bit modules
smte_NEexpver	+7a	2	W	expver from NE header
smte_pathlen	+7c	2	W	length of full pathname
smte_NEexetype	+7e	2	W	exetype from NE header
smte_csegpack	+80	2	W	count of segs to pack

3.6.3 Object Table Entry for OS/2 Warp V3.0

Pointers

SMTE field **smte_objtab** points to the associated OTE for 32-bit modules.

Locations

Dynamically allocated from the kernel swappable heap except for the SMTE associalted with DOSCALLS.DLL.

dcm_ote_start locates the OTE in OS2KRNL associated with DOSCALLS.DLL.

VM Owner

Dynamically allocated OTEs have owner ID **ldrmte (0xffa6)** and share the same heap block as their SMTE.

Format

Field Name	Offset	Length	Type	Description
ote_size	+0	4	D	Object virtual size
ote_base	+4	4	D	Object base virtual address
ote_flags	+8	4	D	Attribute flags
ote_pagemap	+c	4	D	Object page map index
ote_mapsize	+10	4	D	Num of entries in obj page map
ote_resu	+14	4	S	

Table 104. ote_flags Flag Definitions

Name	Bit Mask	Description
OBJREAD	0x00000001L	Readable Object
OBJWRITE	0x00000002L	Writeable Object
OBJEXEC	0x00000004L	Executable Object
OBJRSRC	0x00000008L	Resource Object
OBJDISCARD	0x00000010L	Object is Discardable
OBJSHARED	0x00000020L	Object is Shared
OBJPRELOAD	0x00000040L	Object has preload pages
OBJINVALID	0x00000080L	Object has invalid pages
OBJZEROFIL	0x00000100L	Object has zero-filled pages
OBJRESIDENT	0x00000200L	Object is resident
OBJALIAS16	0x00001000L	16:16 alias required
OBJBIGDEF	0x00002000L	Big/Default bit setting
OBJCONFORM	0x00004000L	Object is conforming for code
OBJIOPL	0x00008000L	Object I/O privilege level
OBJMADEPRIV	0x40000000L	Object is made private for debug (now obsolete)
OBJALLOC	0x80000000L	Object is allocated used by loader

3.6.4 Segment Table Entry for OS/2 Warp V3.0

Pointers
SMTE field **smte_objtab** points to the associated STE for 16-bit modules.

Locations
Dynamically allocated from the kernel swappable heap.

VM Owner
Dynamically allocated STEs have owner ID **ldrmte (0xffa6)** and share the same heap block as their SMTE.

Format

Field Name	Offset	Length	Type	Description
ste_offset	+ 0	2	W	file offset to segment data
ste_size	+ 2	2	W	file data size
ste_flags	+ 4	2	W	type and attribute flags
ste_minsiz	+ 6	2	W	minimum allocation size
ste_seghdl	+ 8	2	W	segment handle
ste_selector	+ a	2	W	segment selector
ste_fixups	+ c	4	D	fixup record storage

Table 105. ste_flags Flag Definitions

Name	**Bit Mask**	**Description**
STE_TYPE_MASK	0x0001	segment type field
STE_CODE	0x0000	code segment type
STE_DATA	0x0001	data segment type
STE_PACKED	0x0002	segment is packed
STE_SEMAPHORE	0x0004	segment semaphore
STE_ITERATED	0x0008	segment data is iterated
STE_WAITING	0x0010	segment is waiting on semaphore
STE_SHARED	0x0020	segment can be shared
STE_PRELOAD	0x0040	segment is preload
STE_ERONLY	0x0080	execute only if code segment read only if data segment
STE_RELOCINFO	0x0100	set if segment has reloc records
STE_CONFORM	0x0200	segment is conforming
STE_RING_2	0x0800	ring 2 selector
STE_RING_3	0x0C00	ring 3 selector
STE_HUGE	0x1000	huge segment
STE_PAGEABLE	0x2000	just a page can be faulted in
STE_PRESENT	0x2000	packed segment already loaded
STE_SELALLOC	0x4000	used to indicate sel allocated
STE_GDTSEG	0x8000	used to indicate GTD sel alloc

3.6.5 Loader Demand Load Data OS/2 Warp V3.0

Pointers

PTDA field **ptda_pldrdldHead** points to chain of modules loaded by **DosLoadModule** for a given process.

_pldrdldHeadKernel points to the head of kernel reference list of LDRDLDs.

Locations

Dynamically allocated from the kernel resident heap.

VM Owner

ldrdld (0xffa4)

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
ldrdld_pldrdldNext	+ 0	4	D	Pointer to next ldrdld
ldrdld_hmte	+ 4	2	W	handle of loaded module
ldrdld_cRef	+ 6	2	W	Number of times loaded

3.7 File System Block Reference

The following control blocks are described in this section:

- 3.7.2, “File System Control Block for OS/2 Warp V3.0” on page 178
- 3.7.3, “System File Table Entry for OS/2 Warp V3.0” on page 182
- 3.7.4, “Master File Table Entry for OS/2 Warp V3.0 ALLSTRICT Kernel” on page 186
- 3.7.5, “Record Lock Record for OS/2 Warp V3.0” on page 188
- 3.7.6, “Volume Parameter Block for OS/2 Warp V3.0” on page 189
- 3.7.7, “Drive Parameter Block for OS/2 Warp V3.0” on page 193
- 3.7.8, “Current Directory Structure for OS/2 Warp V3.0” on page 195
- 3.7.9, “File System Buffer for OS/2 Warp V3.0” on page 197
- 3.7.10, “Named Pipe Structures for OS/2 Warp V3.0” on page 199
- 3.7.11, “Anonymous Pipe Structures for OS/2 Warp V3.0” on page 203

An overview of the file system control blocks is as follows:

3.7.1 File System Control Block Diagrams

The following diagrams illustrate the relationships between various file system control blocks:

- 3.7.1.1, “Open Files - Application to System” on page 173
- 3.7.1.2, “Open Files - System View” on page 174
- 3.7.1.3, “Open Device - System View” on page 175
- 3.7.1.4, “Shared Files with Locked Ranges” on page 176
- 3.7.1.5, “Anonymous and Named Pipes” on page 177

3.7.1.1 Open Files - Application to System

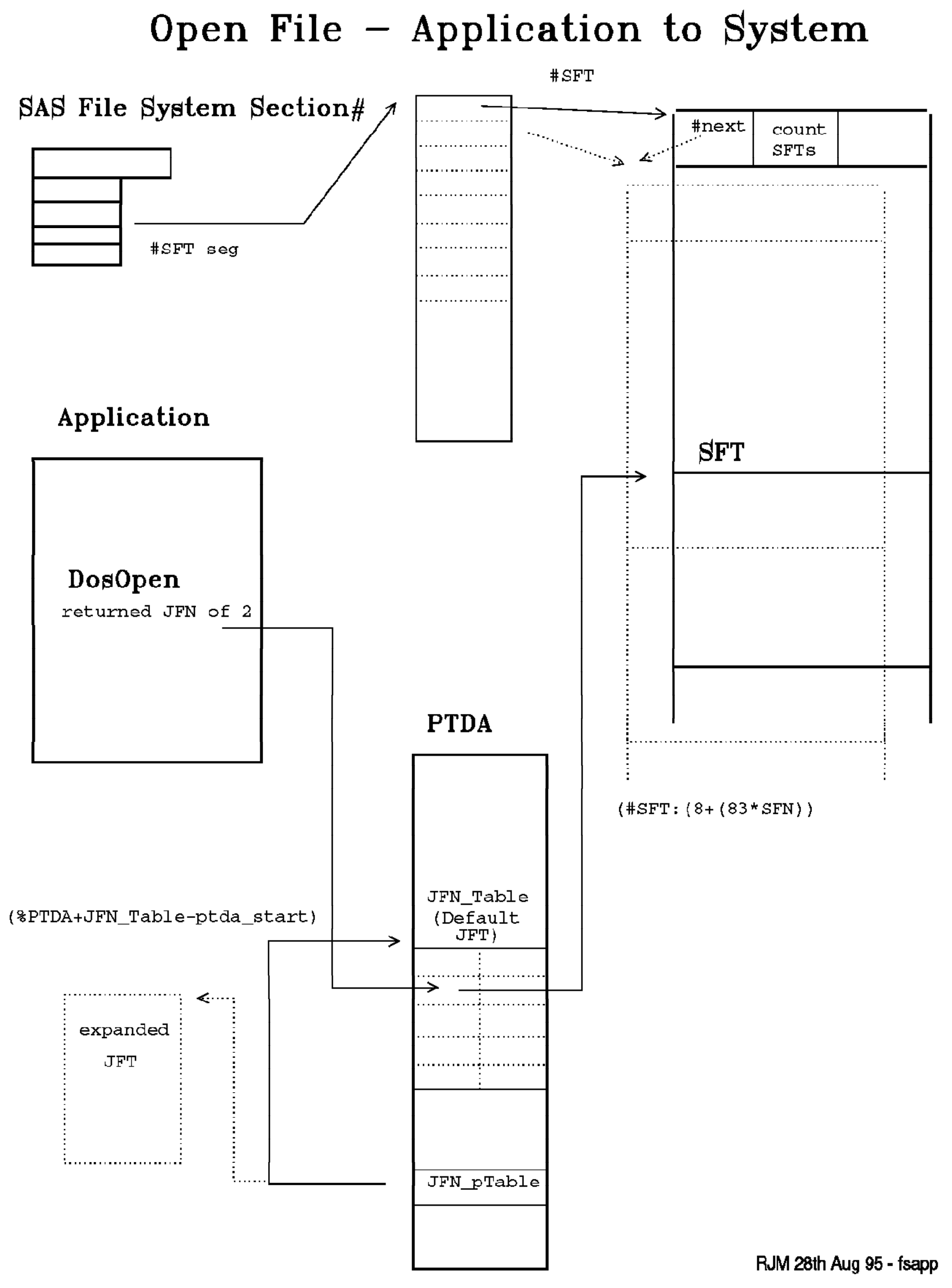


Figure 28. Open Files - Application to System

3.7.1.2 Open Files - System View

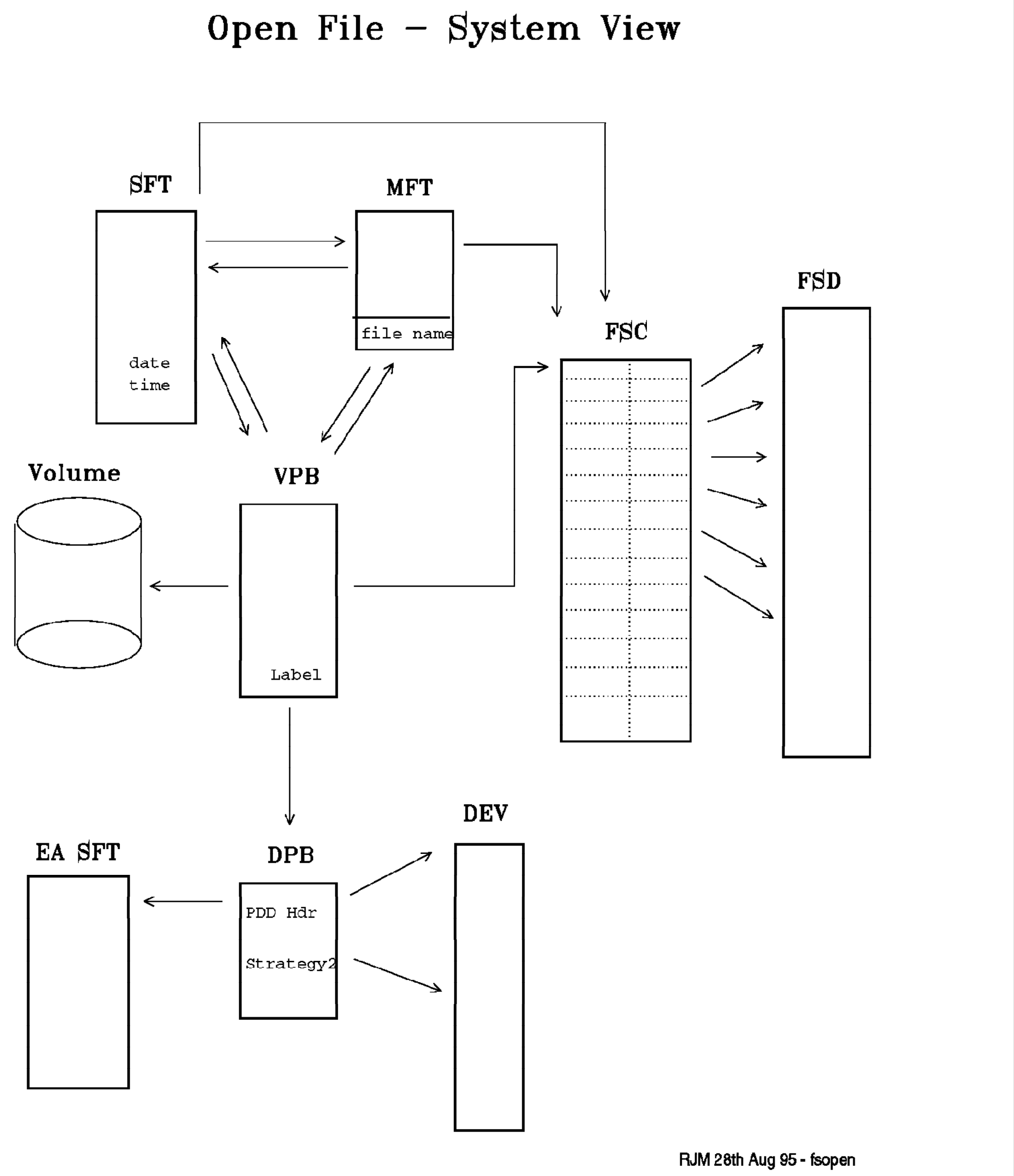


Figure 29. Open Files - System View

3.7.1.3 Open Device - System View

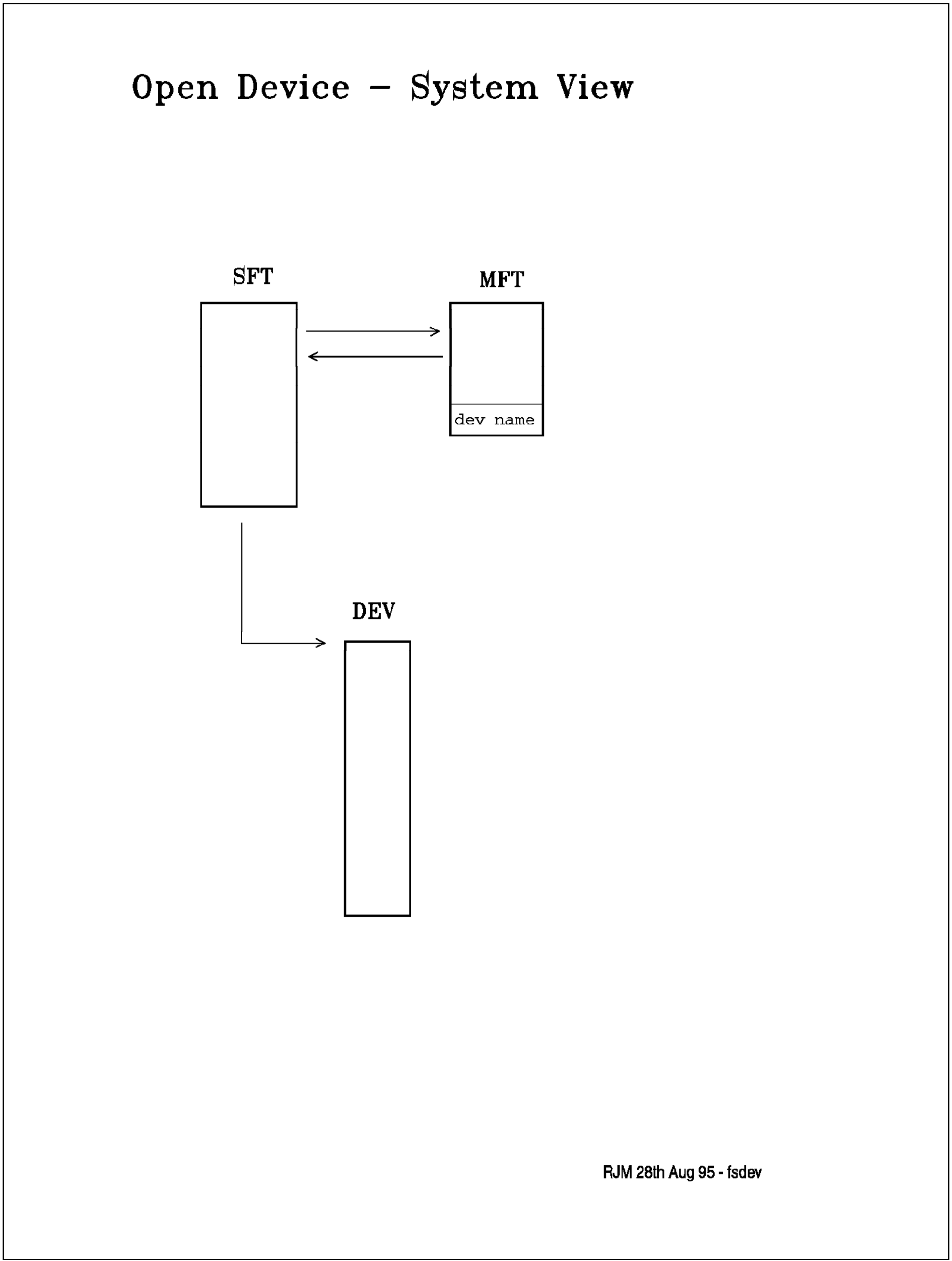


Figure 30. Open Device - System View

3.7.1.4 Shared Files with Locked Ranges

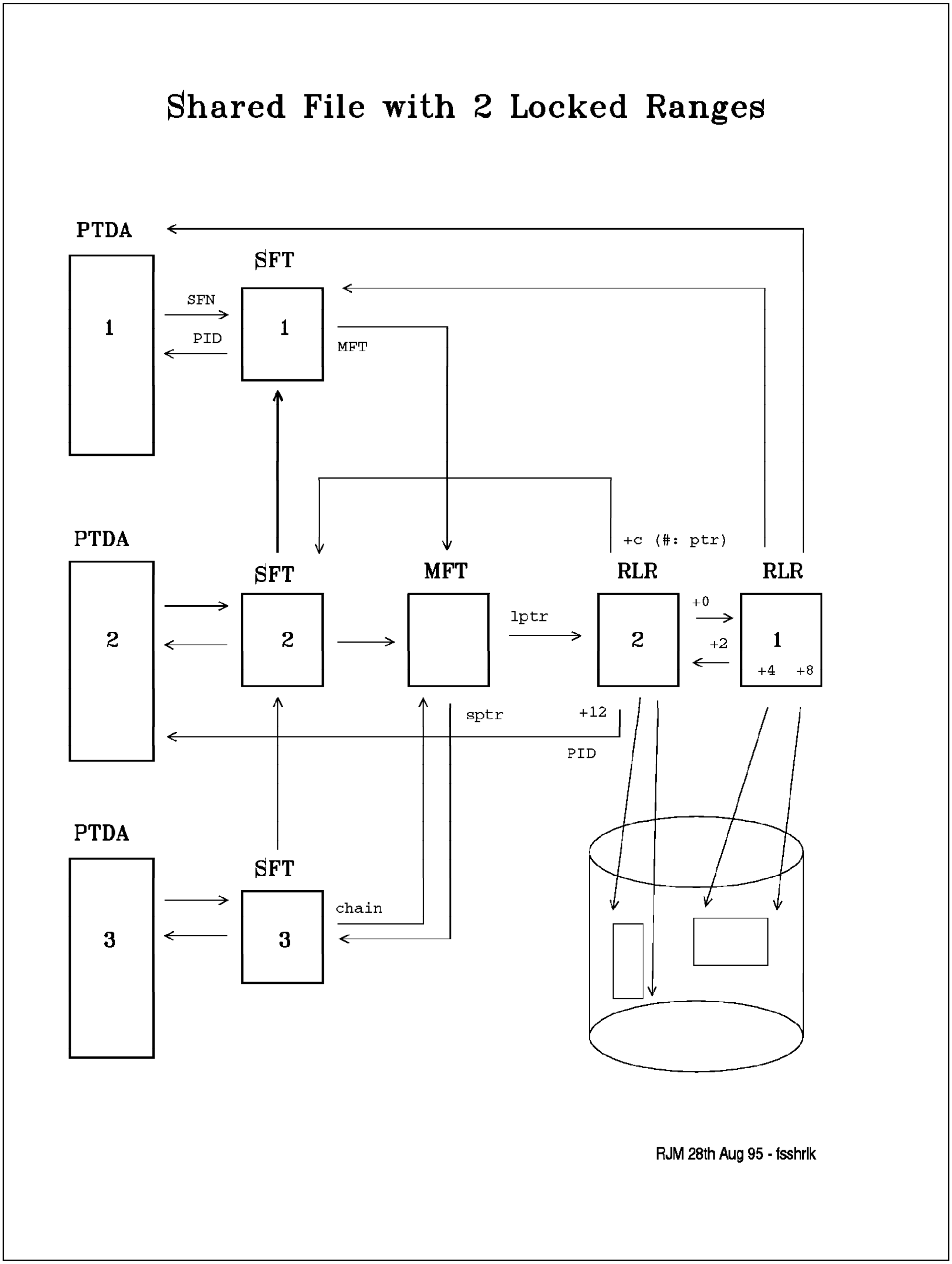


Figure 31. Shared Files with Locked Ranges

3.7.1.5 Anonymous and Named Pipes

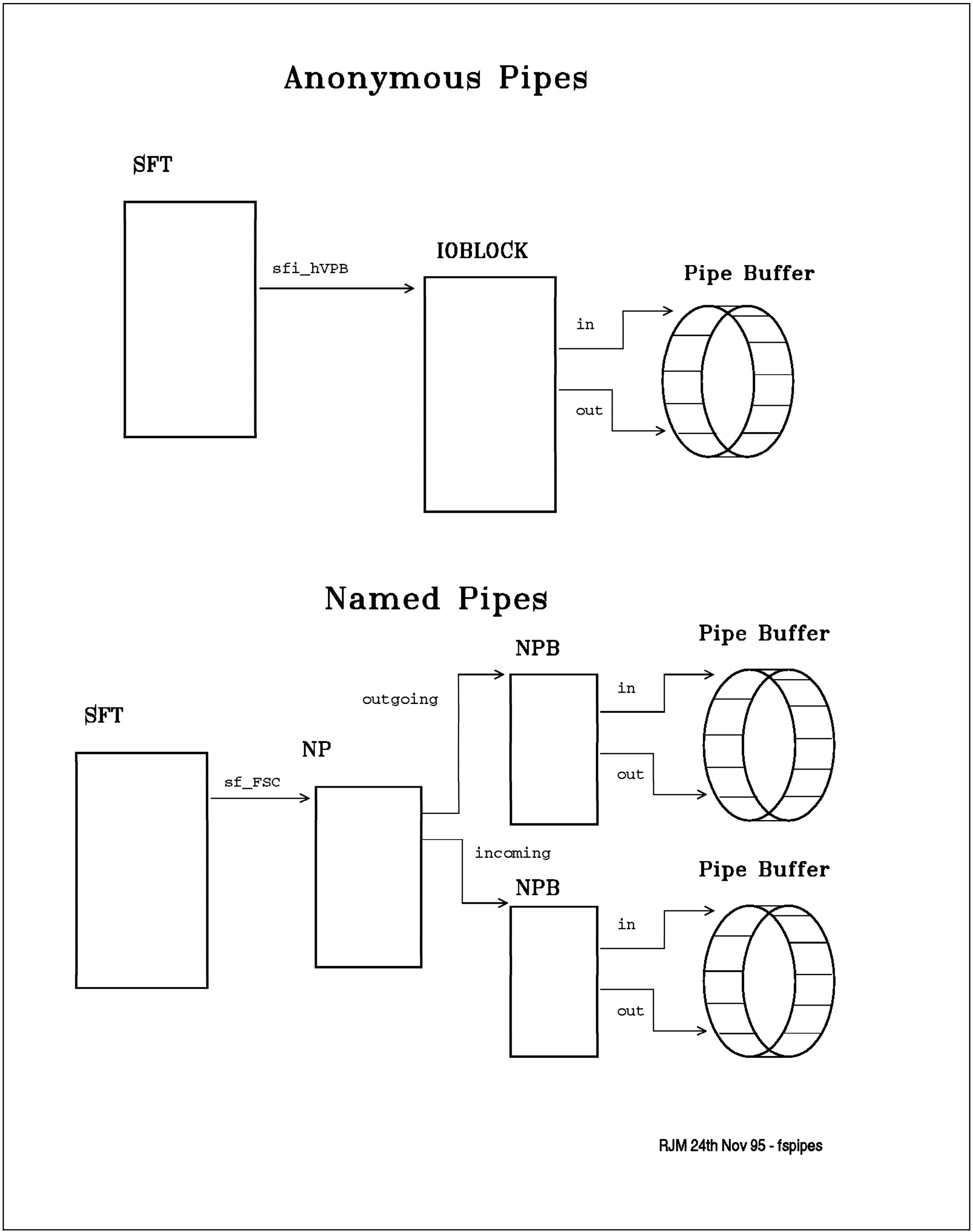


Figure 32. Anonymous and Named Pipes

3.7.2 File System Control Block for OS/2 Warp V3.0

Pointers

SFT field **sf_FSC** points to the associated FSC_ENTRY.

VPB field **vpb_FSC** points to the associated FSC_ENTRY.

CDS field **cd_ownerFSC** points to the associated FSC_ENTRY.

SAS field **SAS_dd_FSC** contains the selector for FSCSEG.

GDT_FSC locates the GDT descriptor for the FSCSEG.

Locations

Dynamically allocated from the kernel resident heap.

VM Owner

fsc (0xff95)

Format

Table 106. FSCSEG

Field Name	Offset	Length	Type	Description
fss_Limit	0	2	W	Offset PAST last allocated byte
fss_ShutdownFlags	2	2	W	flags for shutdown
fss_SDWaitCount	4	2	W	number of processes pending before
fss_pad	6	2	W	shutdown can commence (DWORD align)

Table 107 (Page 1 of 2). FS_ENTRY

Field Name	Offset	Length	Type	Description
FS_ATTRIBUTE	0	4	D	-> FSD attribute. (in FSD memory)
FS_NAME	4	4	D	-> FSD name. (in FSD memory)
FS_ATTACH	8	4	D	DosQFsAttach, DosFsAttach
FS_CHDIR	C	4	D	DosChdir
FS_CHGFILEPTR	10	4	D	DosChgFilePtr
FS_CLOSE	14	4	D	DosClose
FS_COPY	18	4	D	DosCopy
FS_DELETE	1C	4	D	DosDelete
FS_EXIT	20	4	D	DosExit
FS_FILEATTRIBUTE	24	4	D	DosFileInfo, DosSetFileMode
FS_FILEINFO	28	4	D	DosQFileInfo, DosSetFileInfo
FS_FILEIO	2C	4	D	DosFileIO
FS_FINDCLOSE	30	4	D	DosFindClose
FS_FINDFIRST	34	4	D	DosFindFirst

Table 107 (Page 2 of 2). FS_ENTRY

Field Name	**Offset**	**Length**	**Type**	**Description**
FS_FINDFROMNAME	38	4	D	DosFindFromName-Private to server
FS_FINDNEXT	3C	4	D	DosFindNext
FS_FINDNOTIFYCLOSE	40	4	D	DosFindNotifyClose
FS_FINDNOTIFYFIRST	44	4	D	DosFindNotifyFirst
FS_FINDNOTIFYNEXT	48	4	D	DosFindNotifyNext
FS_FSINFO	4C	4	D	DosQFsInfo, DosSetFsInfo
FS_INIT	50	4	D	-- No corresponding API
FS_IOCTL	54	4	D	DosDevIoctl
FS_MKDIR	58	4	D	DosMkdir
FS_MOUNT	5C	4	D	-- No corresponding API
FS_MOVE	60	4	D	DosMove
FS_NEWSIZE	64	4	D	DosNewsize
FS_NMPIPE	68	4	D	All named pipe related API's
FS_OPENCREATE	6C	4	D	DosOpen
FS_PATHINFO	70	4	D	DosQPathInfo, DosSetPathInfo
FS_PROCESSNAME	74	4	D	-- No corresponding API
FS_READ	78	4	D	DosRead, DosReadAsync
FS_RMDIR	7C	4	D	DosRmdir
FS_SETSWAP	80	4	D	-- No correcponding API
FS_WRITE	84	4	D	DosWrite, DosWriteAsync
FS_OPENPAGEFILE	88	4	D	init time only
FS_ALLOCATEPAGESPACE	8C	4	D	size swap file
FS_CANCELLOCKREQUEST	90	4	D	DosCancelLockRequest
FS_FILELOCKS	94	4	D	DosSetFileLocks
FS_VERIFYUNCNAME	98	4	D	Used to save function addresses
FS_COMMIT	9C	4	D	DosBufReset, DosClose
FS_DOPAGEIO	A0	4	D	perform paging
FS_FSCTL	A4	4	D	DosFsCtl
FS_FLUSHBUF	A8	4	D	DosBufReset
FS_SHUTDOWN	AC	4	D	DosShutdown
FS_SDCHGFILEPTR	B0	4	D	Used to save function addresses
FS_SDFSINFO	B4	4	D	at shutdown time. These functions
FS_SDREAD	B8	4	D	will only be called by shutdown
FS_SDWRITE	BC	4	D	filters.

Table 108. FS_ATTRIBUTE Flag Definitions

Name	Bit Mask	Description
FS_ATTR_REMOTE	0x0001	0 = local FSD, 1 = remote FSD
FS_ATTR_UNC	0x0002	0 = normal, 1 = this is UNC FSD
FS_ATTR_LOCKINFO	0x0004	0 = no notice, 1=notify filelocks
FS_ATTR_LVL7	0x0008	0 = no level 7 requests, 1 = yes
FS_ATTR_PIPESVR	0x0010	0 = don′t FSD on PIPE req,1 = yes
FS_ATTR_VERNO	0x7000	bits 28-30 version no
FS_ATTR_EA	0x8000	bit 31 -> 1 = extended attribute

Table 109. FS_COMMIT Flag Definitions

Name	Bit Mask	Description
FS_COMMIT_ALL	2	all handles commit
FS_COMMIT_ONE	1	one handle commit

Table 110. Euqates for Close Type

Name	Bit Mask	Description
FS_CL_ORDINARY	0	ordinary close
FS_CL_FORPROC	1	final close for process
FS_CL_FORSYS	2	final close for system

Table 111. fscnameentstruc

Field Name	Offset	Length	Type	Description
FSCNmEnt_Emulation	0	4	D	
FSCNmEnt_Group	4	1	B	
FSCNmEnt_NameLen	5	1	B	
FSCNmEnt_ProcName	6	1	B	

Table 112. mFS_ENTRY

Field Name	Offset	Length	Type	Description
MFS_CHGFILEPTR	0	4	D	DosChgFilePtr
MFS_CLOSE	4	4	D	DosClose
MFS_INIT	8	4	D	-- No corresponding API
MFS_OPEN	C	4	D	DosOpen
MFS_READ	10	4	D	DosRead, DosReadAsync
MFS_TERM	14	4	D	DosRead, DosReadAsync

Table 113. uncfscentrstruc

Field Name	Offset	Length	Type	Description
UNCfsc_VpbID	0	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	4	4	D	Hold Seg:ofs to UNC FSD′s FSC
UNCfsc_hVPB	8	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	A	2	W	Does this entry contain UNC FSD Info?

Table 114. uncliststruc

Field Name	Offset	Length	Type	Description
UNCfsc_1	0	C	B	
UNCfsc_VpbID	0	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	4	4	D	Hold Seg:ofs to UNC FSD′s FSC
UNCfsc_hVPB	8	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	A	2	W	Does this entry contain UNC FSD Info?
UNCfsc_2	C	C	B	
UNCfsc_VpbID	C	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	10	4	D	Hold Seg:ofs to UNC FSD′s FSC
UNCfsc_hVPB	14	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	16	2	W	Does this entry contain UNC FSD Info?
UNCfsc_3	18	C	B	
UNCfsc_VpbID	18	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	1C	4	D	Hold Seg:ofs to UNC FSD′s FSC
UNCfsc_hVPB	20	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	22	2	W	Does this entry contain UNC FSD Info?
UNCfsc_4	24	C	B	
UNCfsc_VpbID	24	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	28	4	D	Hold Seg:ofs to UNC FSD′s FSC
UNCfsc_hVPB	2C	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	2E	2	W	Does this entry contain UNC FSD Info?

3.7.3 System File Table Entry for OS/2 Warp V3.0

Pointers

MFT field **mft_sptr** points to the associated sf_entry.

RLR field **rlr_sptr** points to the associated sf_entry.

SAS field **SAS_file_SFT** contains the selector for the SFT segment table.

NP field **np_ssft** points to the server SFT for a named pipe.

NP field **np_csft** points to the client SFT for a named pipe.

GDT_SFT locates the GDT descriptor for the SFT segement table.

Locations

Dynamically allocated from the system arena.

VM Owner

sft (0xffa1).

Format

Table 115. SFT

Field Name	**Offset**	**Length**	**Type**	**Description**
sft_link	0	2	W	selector for next chunk of table
sft_count	2	2	W	number of entries in this block
sft_handle	4	2	W	handle of segment holding this block
sft_inshutdown	6	2	W	flags for shutdown

Table 116 (Page 1 of 3). sf_entry

Field Name	**Offset**	**Length**	**Type**	**Description**
sf_ref_count	0	2	W	number of processes sharing entry
sf_usercnt	2	2	W	For files: number of threads waiting for access to sf_entry. For devices: number of threads using this sf_entry.
reserved	4	1	B	Used to be attr of file - moved to * independent part of the SFT for general * access

Table 116 (Page 2 of 3). sf_entry

Field Name	**Offset**	**Length**	**Type**	**Description**
sf_flags	5	2	W	Bits 8-15 Bit 15 = 1 if remote file = 0 if local file or device Bit 14 = not used Bit 13 = Pipe bit Bit 12 = FCB bit = 1 if fcb sft = 0 if normal sft Bit 11 = if Pipe, = 0 if anonymous pipe = 1 if named pipe Bit 10 == sf_inuse = sf_entry is in use by some thread, ie busy Bit 9 == sf_want = some thrd blocked waiting to use the sf_entry Bit 8 == sf_noJFN, no handle alloced for sft Bits 0-7 (old FCB_devid bits) If remote file or local file, bit 6=0 if dirty Device ID number, bits 0-5 if local file. bit 7=0 for local file =1 for local I/O device If local I/O device, bit 6=0 if EOF (input) Bit 5=1 if Raw mode Bit 0=1 if console input device Bit 1=1 if console output device Bit 2=1 if null device Bit 3=1 if clock device
sf_flags2	7	2	W	
sf_devptr	9	4	D	Not used if local file, points
sf_FSC	D	4	D	Pointer to the file system control
sf_cookie	11	4	D	server's per-file id (for oplock support)
sf_chain	15	4	D	16:16 Link to next SFT
sf_MFT	19	4	D	32-bit FLAT pointer to MFT entry

Table 116 (Page 3 of 3). sf_entry

Field Name	Offset	Length	Type	Description
sf_fsd	1D	32	S	File system dependent section
sf_fsi	4F	30	S	File system independent section
sf_plock	7F	2	W	16-bit offset to first pending LOCK record
sf_NmPipeSfn	81	2	W	SFN of named pipe for spooled files
sf_codepage	83	2	W	current codepage (font) for data in file
sf_LockID	85	4	D	lock-id for protected file-handle access

Table 117. sf_flags Flag Definitions

Name	Bit Mask	Description
SF_ISNET	0x8000	True if SFT is for remote
SF_PIPE	0x2000	Anonymous Pipe
SF_FCB	0x1000	True if SFT is for an FCB
SF_NMPIPE	0x0800	true if name pipe
SF_INUSE	0x0400	True if sf entry is in use by some thread, that is, busy
SF_BLOCKED	0x0200	True if some thread is blocked waiting to use the sf entry
SF_NOJFN	0x0100	True if no handle alloc′ed for SFT

Table 118 (Page 1 of 2). sf_flags2 Flag Definitions

Name	Bit Mask	Description
SF_FORMAT_MOUNT	0x8000	True if a format mount was done, and still in effect
SF_BEGINFORMAT_FAILED	0x4000	True if a beginformat ioctl failed
SSF2_LDRBINARYSEM	0x2000	′ON′ if SFT owned by some thread
SF_SRVRDR	0x1000	serving pipe redirection in effect
SFF2_LOCKED_DRIVE	0x0800	A LOCK was issued on this direct access handle to lock the drive.
SFF2_SPOOLED	0x0400	File is spooled
SFF2_DATAWRITTEN	0x0200	Data written to file
SFF2_Consistency	0x0180	consistency bits
SFF2_CANCELJOB	0x0040	spool job has been canceled*/ ;whs
SFF2_NONSPOOLED	0x0020	File is nonspooled; going to printer
SFF2_STPTHINFDN	0x0010	SetPathInfo done, don′t set archive
sff2_RA_ON	0x0008	Readahead started

Table 118 (Page 2 of 2). sf_flags2 Flag Definitions

Name	**Bit Mask**	**Description**
sff2_UNC	0x0004	UNC object
sff2_isfree	0x0002	this SFT is on free list (unused)
sff2_RA_BIG	0x0001	Big Readahead

Table 119. sfdFATFS

Field Name	**Offset**	**Length**	**Type**	**Description**
sfdFAT_firFILEclus	0	2	W	First cluster of file (bit 15 = 0)
sfdFAT_cluspos	2	2	W	Position of last cluster accessed
sfdFAT_lstclus	4	2	W	Last cluster accessed
sfdFAT_dirsec	6	4	D	Sector # of directory sector for this file
sfdFAT_dirpos	A	1	B	Offset of this entry in the above
sfdFAT_EAHandle	B	2	W	starting cluster of EAs
sfdFAT_name[11]	D	B	B	
sfdFAT_bRAReads	18	4	D	# of consecutive reads within range
sfdFAT_bRABigReads	1C	4	D	# of consecutive big reads
sfdFAT_fldMask	20	4	D	Unique File Dirty Mask
sfdFAT_pSFT	24	4	D	Linear address of SFT
sfdFAT_ulNextRA	28	4	D	Position where next rahead starts
sfdFAT_bBufRun	2C	4	D	Number of sectors in rahead run
sfdFAT_LastFATSec	30	2	W	last FAT sector added to chain

Table 120. sftfsd

Field Name	**Offset**	**Length**	**Type**	**Description**
sfd_work[50]	0	32	B	

Table 121 (Page 1 of 2). sftsfi

Field Name	**Offset**	**Length**	**Type**	**Description**
sfi_mode	0	2	W	mode of access or high bit on if FCB
sfi_mode2	2	2	W	additional openmode bits for DosOpen2
sfi_hVPB	4	2	W	handle of volume
sfi_ctime	6	2	W	Creation time of file

Table 121 (Page 2 of 2). sftsfi

Field Name	**Offset**	**Length**	**Type**	**Description**
sfi_cdate	8	2	W	Creation date of file
sfi_atime	A	2	W	Time of last access of file
sfi_adate	C	2	W	Date of last access of file
sfi_mtime	E	2	W	Time of last modification of file
sfi_mdate	10	2	W	Date of last modification of file
sfi_size	12	4	D	Size associated with file
sfi_position	16	4	D	Read/Write pointer or LRU count for FCBs
sfi_UID	1A	2	W	User ID
sfi_PID	1C	2	W	Process ID
sfi_PDB	1E	2	W	Process Data Block
sfi_selfsfn	20	2	W	SFN of this sf_entry, used to speed
sfi_tstamp	22	1	B	update time stamp flags; see ST_ equs
sfi_type	23	2	W	file/device/named-pipe/FCB
pPVDBFil	25	4	D	performance counter data block pointer
sfi_DOSattr	29	1	B	DOS attributes of file(sys,hid,r/o,arch

Table 122. sfi_type Flag Definitions

Name	**Bit Mask**	**Description**
STYPE_FILE	0x0000	file
STYPE_DEVICE	0x0001	device
STYPE_NMPIPE	0x0002	named pipe
STYPE_FCB	0x0004	SFT is for an FCB

3.7.4 Master File Table Entry for OS/2 Warp V3.0 ALLSTRICT Kernel

For **MTE** formats for other versions of OS/2 see:

3.7.4.1, “Master File Table Entry for OS/2 Warp V3.0 RETAIL Kernel” on page 188

Pointers

SFT field **sf_MFT** points to the associated MFT entry.

SAS field **SAS_file_MFT** points to the PTREE head for the MFT PTREE.

Locations

Dynamically allocated from the kernel resident heap.

VM Owner
mft (0xff9e).

Format

Field Name	Offset	Length	Type	Description
mft_ksem	+0	10	S	multi read/single write semaphore
mft_lptr	+10	2	W	16-bit offset to first LOCK record
mft_sptr	+12	4	D	16-bit FAR pointer to first SFT in chain
mft_pCMap	+16	4	D	32-bit FLAT pointer to cluster map
mft_CMapKSem	+1a	10	S	semaphore for access to pCMap
mft_opflags	+2a	2	W	oplock flags
mft_serl	+2c	2	W	serial number for FCB checking
mft_flags	+2e	2	W	general purpose MFT flags
mft_signature	+30	2	W	for sanity check
mft_hvpb	+32	2	W	handle of vpb
mft_name	+34	1	B	start of name string (zero terminated)

Table 123. mft_flags Flag Definitions

Name	Bit Mask	Description
mft_pagerheap	0x0001	MFT is allocated on pager heap
MFT_DEFAULTHEAP	0x0	MFT is allocated on kernel (heap default MFT heap)

Table 124. mft_opflags Flag Definitions

Name	Bit Mask	Description
mft_opnolock	0	no oplock or opbatch on file
mft_oplock	1	oplock on file
mft_opbatch	2	opbatch on file
mft_opbreak	4	oplock/batch cleanup in process
mft_opbreakfailed	8	oplock/batch cleanup failed

3.7.4.1 Master File Table Entry for OS/2 Warp V3.0 RETAIL Kernel

Table 125. Master File Table Entry for OS/2 Warp V3.0 RETAIL Kernel

Field Name	**Offset**	**Length**	**Type**	**Description**
mft_ksem	+0	C	S	multi read/single write semaphore
mft_lptr	+c	2	W	16-bit offset to first LOCK record
mft_sptr	+e	4	D	16-bit FAR pointer to first SFT in chain
mft_pCMap	+12	4	D	32-bit FLAT pointer to cluster map
mft_CMapKSem	+16	C	S	semaphore for access to pCMap
mft_opflags	+22	2	W	oplock flags
mft_serl	+24	2	W	serial number for FCB checking
mft_flags	+26	2	W	general purpose MFT flags
mft_hvpb	+28	2	W	handle of vpb
mft_name	+2a	1	B	start of name string (zero terminated)

3.7.5 Record Lock Record for OS/2 Warp V3.0

Pointers

MFT field **mft_lptr** contains the offset within the RLR segment to the first RLR associated with the MFT.

GDT_FSC locates the GDT descriptor for the RLR segment.

Locations

Dynamically allocated from the system arena.

VM Owner

fsreclok (0xff47).

Format

Field Name	**Offset**	**Length**	**Type**	**Description**
rlr_next	+0	2	W	16-bit offset to next RLR. 0 if end
rlr_prev	+2	2	W	16-bit offset to prev RLR. 0 if SFT
rlr_fba	+4	4	D	offset of first byte of locked region
rlr_lba	+8	4	D	offset of last byte of locked region

Field Name	Offset	Length	Type	Description
rlr_sptr	+c	4	D	16:16 FAR pointer to SFT
rlr_UID	+10	2	W	lock issuer′s user ID
rlr_PID	+12	2	W	lock issuer′s process ID
rlr_PDB	+14	2	W	lock issuer′s PDB, 0 for non-3xBox
rlr_flags	+16	1	B	flags

rlr_flags flag definitions

Table 126. rlr_flags Flag Definitions

Name	Bit Mask	Description
RLR_EXCLUSIVE	0	
RLR_SHARED	1	
RLR_WAITING	2	
RLR_CANCELPLOCK	4	

3.7.6 Volume Parameter Block for OS/2 Warp V3.0

Pointers

SFT field **sfi_hVPB** contains the offset within the VPB segment of the associated VPB.

MFT field **mft_hVPB** contains the offset within the VPB segment of the associated VPB.

DPB field **dpb_hVPB** contains the offset within the VPB segment of the associated VPB.

CDS field **cdi_hVPB** contains the offset within the VPB segment of the associated VPB.

GDT_VPB locates the GDT descriptor for the VPB segment.

Locations

VPB segment is dynamically allocated from the kernel resident heap.

VM Owner

vpb (0xffa2).

Format

vpb

Table 127 (Page 1 of 2). vpb Format

Field Name	Offset	Length	Type	Description
vpb_flink	0	2	W	handle of forward link

Table 127 (Page 2 of 2). vpb Format

Field Name	Offset	Length	Type	Description
vpb_blink	2	2	W	handle of back link
vpb_ref_count	4	2	W	count of objects that point to VPB
vpb_search_count	6	2	W	count of searches that point to VPB
vpb_first_access	8	1	B	initialized to -1 to force a media
vpb_signature	9	2	W	Signature specifying VPB validity
vpb_flags	B	1	B	flags (bits 7,6,3-0 defined below)
vpb_fMisc	C	1	B	More flags (bit 7 defined below)
vpb_FSC	D	4	D	Pointer to the file system control block (FSC).
vpb_fsd	11	40	S	File system dependent section
vpb_fsi	51	2C	S	File system independent section

Table 128. vpb_signature Values

Name	Bit Mask	Description
VPB_VALID	0x444A	
VPB_INVALID	0x4A47	

Table 129. vpb_ID Values

Name	Bit Mask	Description
UNREAD_ID1	0x4A52	Media unreadable
UNREAD_ID2	0x534E	Media unreadable
DAMAGED_ID1	0x0000	Media damaged but recognised by IFS
DAMAGED_ID2	0x0000	Media damaged but recognised by IFS

Table 130 (Page 1 of 2). vpb_falgs Flag Definitions

Name	Bit Mask	Description
VPBCHECK	0x01	a volume ID check is going on for this VPB
VPBNEWBOOT	0x02	new format disk
VPBMOUNT	0x04	Mount in progress
VPBFORMATMOUNT	0x08	FormatMount done, not cleared
VPBINVALID	0x10	volume formatted - old vpb invalid
VPBINITCACHE	0x20	Initializing Cache Data

Table 130 (Page 2 of 2). vpb_falgs Flag Definitions

Name	Bit Mask	Description
VPBSETVID	0x40	vid set is in progress
VPBALLOCATE	0x80	cluster allocation in progress

Table 131. vpb_fMisc Flag Definitions

Name	Bit Mask	Description
VPB_FM_WRITEABLE	0x01	Set if we know volume can be written
VPB_FM_UNKNOWN	0x02	Set if no FATs and not claimed by FSD
VPB_REMOTE_DRIVE	0x04	set for attaches of remote drives
VPB_FM_ALLOCSHWAIT	0x08	Set if somebody wants alloc access so that they can get some disk clusters for this volume
VPB_FM_ALLOCEXWAIT	0x10	excl.access wait for somebody who wants to release some clusters
VPB_FM_INITCACHE_ERROR	0x20	Error initializing cache
VPB_FM_INITCACHE_DONE	0x40	

Table 132. vpbfsd Format

Field Name	Offset	Length	Type	Description
vpd_work[64]	0	40	B	

Table 133. vpbfsi Table

Field Name	Offset	Length	Type	Description
vpi_ID	0	4	D	32-bit unique ID of file (See UNREAD_IDx, DAMAGED_IDx)
vpi_pDPB	4	4	D	Drive volume is in
vpi_cbSector	8	2	W	Size of physical sector in bytes
vpi_totsec	A	4	D	Total number of sectors on medium
vpi_trksec	E	2	W	Sectors per track on medium
vpi_nhead	10	2	W	Number of heads in device
vpi_text[12]	12	C	B	
vpi_pDCS	1E	4	D	device capability struc
vpi_pVCS	22	4	D	volume characteristic struc
vpi_drive	26	1	B	drive (0=A)
vpi_unit	27	1	B	unit
vpi_flags	28	2	W	flags for memory restrictions

Table 134. vpdFATFS Table

Field Name	**Offset**	**Length**	**Type**	**Description**
vpdFAT_cluster_mask	0	1	B	Sectors/cluster - 1
vpdFAT_cluster_shift	1	1	B	Log2 of sectors/cluster
vpdFAT_first_FAT	2	2	W	Starting record of FATs
vpdFAT_FAT_count	4	1	B	Number of FATs for this drive
vpdFAT_root_entries	5	2	W	Number of directory entries
vpdFAT_first_sector	7	4	D	First sector of first cluster
vpdFAT_max_cluster	B	2	W	Number of clusters on drive + 1
vpdFAT_FAT_size	D	2	W	Number of records occupied by FAT
vpdFAT_dir_sector	F	4	D	Starting record of directory
vpdFAT_media	13	1	B	Media byte (duplicate of VPB)
vpdFAT_next_free	14	2	W	Cluster # of last allocated cluster
vpdFAT_free_cnt	16	2	W	Count of free clusters, -1 if unknown
vpdFAT_FATentrysize	18	1	B	12 or 16 - can you guess why ??? @@
vpdFAT_IDsector	19	4	D	sector number of ID
vpdFAT_minEOF	1D	2	W	minimum EOF cluster value: 12-bit -> FF8, 16-bit -> FFF8
vpdFAT_access	1F	2	W	whether rmdir XOR mov dir XOR (chdir mkdir OR mov file OR create)* has access to volume
vpdFAT_accwait	21	2	W	who′s waiting for access
vpdFAT_alloc	23	2	W	whether disk cluster alloc OR release
vpdFAT_eaflags	25	2	W	flags for EA usage
vpdFAT_eareaders	27	2	W	number of threads with pending reads
vpdFAT_eawaiters	29	2	W	number of threads waiting to run
vpdFAT_eahandles	2B	2	W	number of handles in EAOffTable
vpdFAT_pEASFT	2D	4	D	SFT for ″EA DATA. SF″
vpdFAT_pBadSector	31	4	D	Ptr for Bad sectors data
vpdFAT_pClusBitMap	35	4	D	Ptr to free cluster bit map
vpdFAT_pNextFreeBitMap	39	4	D	Ptr to next free bit map position
vpdFAT_cNextFreeBitMap	3D	2	W	Count of dwords remaining in bit map

Name	Bit Mask	Description
eavpb_fileopen	0x0001	the EA file on this volume is open
eavpb_changing	0x0002	the EA file is changing
eavpb_dooropen	0x0004	the drive door has been opened

Table 135. vpdFAT_eaflags Flag Definitiions

3.7.7 Drive Parameter Block for OS/2 Warp V3.0

Pointers

VPB field **vpi_pDPB** points to the associated DPB.

Locations

DPB segment is dynamically allocated from the kernel resident heap.

VM Owner

dpb (0xff96).

Format

Table 136. DPB Format

Field Name	Offset	Length	Type	Description
dpb_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)
dpb_unit	+1	1	B	Driver unit number of DPB
dpb_driver_addr	+2	4	D	Pointer to driver
dpb_next_dpb	+6	4	D	Pointer to next Drive parameter block
dpb_cbSector	+a	2	W	sector size (for volume checking)
dpb_first_FAT	+c	2	W	sector of 1st FAT (for ancient dev drivers)
dpb_toggle_time	+e	4	D	time of last drive toggle
dpb_hVPB	+12	2	W	handle of volume currently in drive
dpb_media	+14	1	B	most recent media that was in drive
dpb_flags	+15	1	B	synchronization flags (see below)
dpb_drive_lock	+16	2	W	Contains pid if drive locked by process
dpb_strategy2	+18	4	D	strategy2 addr (or 00000000)

Table 137. dpb_flags Flag Definitions		
Name	**Bit Mask**	**Description**
DPBCHECK	0x10	disk in drive is being removed/checked for VPB
DPBNONREMOV	0x20	1 => drive supports non-removable media
DPBVCRAMDISK	0x40	Ram Disk Driver

Table 138. DPB3X Table				
Field Name	**Offset**	**Length**	**Type**	**Description**
dpb3x_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)
dpb3x_UNIT	+1	1	B	Driver unit number of DPB
dpb3x_sector_size	+2	2	W	Size of physical sector in bytes
dpb3x_cluster_mask	+4	1	B	Sectors/cluster - 1
dpb3x_cluster_shift	+5	1	B	Log2 of sectors/cluster
dpb3x_first_FAT	+6	2	W	Starting record of FATs
dpb3x_FAT_count	+8	1	B	Number of FATs for this drive
dpb3x_root_entries	+9	2	W	Number of directory entries
dpb3x_first_sector	+b	2	W	First sector of first cluster
dpb3x_max_cluster	+d	2	W	Number of clusters on drive + 1
dpb3x_FAT_size	+f	1	B	Number of records occupied by FAT
dpb3x_dir_sector	+10	2	W	Starting record of directory
dpb3x_driver_addr	+12	4	D	Pointer to driver
dpb3x_media	+16	1	B	Media byte
dpb3x_first_access	+17	1	B	This is initialized to -1 to force a media check the first time this DPB is used
dpb3x_next_dpb	+18	4	D	Pointer to next Drive parameter block
dpb3x_next_free	+1c	2	W	Cluster # of last allocated cluster
dpb3x_free_cnt	+1e	2	W	Count of free clusters, -1 if unknown

Table 139 (Page 1 of 2). DPB4X Table				
Field Name	**Offset**	**Length**	**Type**	**Description**
dpb4x_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)

Table 139 (Page 2 of 2). DPB4X Table

Field Name	Offset	Length	Type	Description
dpb4x_UNIT	+1	1	B	Driver unit number of DPB
dpb4x_sector_size	+2	2	W	Size of physical sector in bytes
dpb4x_cluster_mask	+4	1	B	Sectors/cluster - 1
dpb4x_cluster_shift	+5	1	B	Log2 of sectors/cluster
dpb4x_first_FAT	+6	2	W	Starting record of FATs
dpb4x_FAT_count	+8	1	B	Number of FATs for this drive
dpb4x_root_entries	+9	2	W	Number of directory entries
dpb4x_first_sector	+b	2	W	First sector of first cluster
dpb4x_max_cluster	+d	2	W	Number of clusters on drive + 1
dpb4x_FAT_size	+f	2	W	Number of records occupied by FAT
dpb4x_dir_sector	+11	2	W	Starting record of directory
dpb4x_driver_addr	+13	4	D	Pointer to driver
dpb4x_media	+17	1	B	Media byte
dpb4x_first_access	+18	1	B	This is initialized to -1 to force a media check the first time this DPB is used
dpb4x_next_dpb	+19	2	D	Pointer to next Drive parameter block
dpb4x_next_free	+1d	2	W	Cluster # of last allocated cluster
dpb4x_free_cnt	+1f	2	W	Count of free clusters, -1 if unknown

3.7.8 Current Directory Structure for OS/2 Warp V3.0

Pointers

SAS field **SAS_file_CDS** contains the selector for CDS RMP segment.

CDSAddr locates the RMP handle which contains the selector for the CDS RMP segment.

Locations

CDS segment is dynamically allocated from the kernel resident heap.

VM Owner

cdsrmp (0xff61).

Format

Table 140. cddFATFS Table

Field Name	Offset	Length	Type	Description
cddFAT_id	0	2	W	cluster of current dir

Table 141. cdfsd Table

Field Name	Offset	Length	Type	Description
cdd_work[8]	0	8	S	

Table 142. cdfsi Table

Field Name	Offset	Length	Type	Description
cdi_hVPB	0	2	W	hVPB for the drive mapped to this CDS
cdi_end	2	2	W	End of assignment
cdi_flags	4	1	B	fs independent flags (see below)
cdi_text[260]	5	104	A	

Table 143. curdir Table

Field Name	Offset	Length	Type	Description
cd_handle	0	2	W	lookup key for this CDS
cd_pid	2	2	W	PID part of lockup key for handles 1-26
cd_refcnt	4	2	W	reference count CDSs
cd_flags	6	1	B	See below for definitions
cd_devptr	7	4	D	local pointer to DPB or net device
cd_OwnerFSC	B	2	W	Owner FSC.Offst
cd_fsd	D	8	S	File system dependent section
cd_fsi	15	10A	S	File system independent section
cdi_hVPB	15	2	W	hVPB for the drive mapped to this CDS
cdi_end	17	2	W	End of assignment
cdi_flags	19	1	B	fs independent flags (see below)
cdi_text[260]	1A	104	B	
	11E	1	B	

Table 144 (Page 1 of 2). cd_flags Flag Definitions

Name	Bit Mask	Description
CD_ISNET	0x80	This CDS is for a remote drive

Table 144 (Page 2 of 2). cd_flags Flag Definitions

Name	**Bit Mask**	**Description**
CD_INUSE	0x40	This CDS is in use
CD_SPLICE	0x20	This CDS is for a JOINed drive
CD_JOIN	CD_SPLICE	This CDS is for a JOINed drive
CD_LOCAL	0x10	This CDS is for a SUBSTed drive
CD_ISPSEUDOCHAR	0x08	This CDS for a pseudo-char dev
CD_ISUNC	0x04	This CDS for a UNC name

Table 145. cdi_flags Flag Definitions

Name	**Bit Mask**	**Description**
CDI_ISVALID	0x80	This CDS contains a valid cd_fsd
CDI_ISROOT	0x40	This CDS is for a root (no cdfsd)
CDI_MEDIASWAPPED	0x20	This CDS may not be valid (forces

3.7.9 File System Buffer for OS/2 Warp V3.0

Pointers

SAS field **SAS_file_Buffers** contains the selector for the file system buffer segment.

GDT_Buffers locates the GDT descriptor for the BUFSEG segment.

Locations

BUFSEG segment is dynamically allocated from the kernel resident heap.

VM Owner

fsbuf (0xff93).

Format

Table 146 (Page 1 of 2). BUFSEG Format

Field Name	**Offset**	**Length**	**Type**	**Description**
bs_MRUHead	+ 0	2	W	Head of MRU buffer list (LRU tail)
bs_MRUTail	+ 2	2	W	Tail of MRU buffer list (LRU head)
bs_FreeHead	+ 4	2	W	Head of Free buffer list
bs_Handle	+ 6	2	W	Handle for virtual memory manager

Table 146 (Page 2 of 2). BUFSEG Format

Field Name	**Offset**	**Length**	**Type**	**Description**
bs_nBuffers	+ 8	2	W	Number of buffers in segment
bs_buffsize	+ a	2	W	Size of buffer+header, in bytes.
bs_seglimit	+ c	2	W	Limit for entire buffer segment.
bs_pStats	+ e	2	W	Offset of statistics block (for PROFILE)
bs_offRemMed	+10	2	W	Minimum ″legal″ offset of buffer for removable media
bs_MaxSec	+12	2	W	Maximum sector size for block device drivers
bs_BigBufBase	+14	2	W	Base of big buffers pool
bs_BigBufMap	+16	2	W	Big buffers usage bit map (bit0 - Buf0)
bs_physBufSeg	+18	4	D	Buffer segment Physical Address

Table 147. BUFFINFO Format

Field Name	**Offset**	**Length**	**Type**	**Description**
buf_next	+ 0	2	W	Pointer to next buffer in list (-1 = end)
buf_prev	+ 2	2	W	Pointer to previous buffer in list (-1 = end)
buf_freeLink	+ 4	2	W	Pointer to next free buffer (-1 = end)
buf_hVPB	+ 6	1	W	serial number of volume
buf_sector	+ 8	4	D	Sector number of buffer
buf_wrtcnt	+ c	1	B	For FAT sectors, # times sector written out
buf_wrtcntinc	+ d	2	W	For FAT sectors, # sectors between each write
buf_flags	+ f	1	B	Flags
buf_tid	+10	2	W	thread ID of buffer owner
buf_refcnt	+12	2	W	number of threads using buffer for read
buf_fill	+14	2	W	random debugging information
buf_pad	+16	2	W	Force dword alignment.

Table 148. buf_falgs Flag Definitions

Name	**Bit Mask**	**Description**
BUF_DIRTY	0x80	Bit 7 = 1 if buffer dirty
BUF_VISIT	0x40	Bit 6 = 1 if buffer seen in search
BUF_WANT	0x20	Bit 5 = 1 if process waiting for buffer
BUF_BUSY	0x10	Bit 4 = 1 if in use by process
BUF_ISDATA	0x08	Bit 3 = 1 if buffer is DATA
BUF_ISDIR	0x04	Bit 2 = 1 if buffer is DIR
BUF_ISFAT	0x02	Bit 1 = 1 if buffer is FAT
BUF_ATTEMPTING_READ	0x01	Bit 0 = 1 if buffer is in swapper pool

3.7.10 Named Pipe Structures for OS/2 Warp V3.0

Pointers

SFT field **sf_fsd** points to the associated NP structure.

NmpRmpHand locates the RMP handle that contains the selector for the NPN RMP segment.

NPN field **npn_link** points to the list of linked NP structures.

NP fields **np_selector1** and **np_selector2** point to associated NPB structures.

Locations

The NPN RMP is allocated from the kernel swappable heap.

The NP is allocated from the system arena.

The NPB is allocated from the kernel resident heap.

VM Owner

NP owner ID is **npipenp (0xff31)**.

NPN owner ID is **npipenpn (0xff30)**.

NPB owner ID is **npipenbuf (0xff9f)**.

Format

There are four important data structures associated with named pipes: the SFT corresponding to an open named pipe, a pair of kernel internal data structures describing the pipe and one or two allocated memory segments which contain the data buffers for the pipe.

The parts of the SFT specific to named pipes are:

```
sf_flags       SF_NMPIPE and SF_PIPE set
sf_np          pointer to pipe info.
sf_pipmod      mode of pipe, per-sft internal state bits
```

```
Where:

    sf_np is defined to be sf_fsd+0, the pointer to np structure
    sf_pipmod is defined to be sf_fsd+4, the mode of pipe, plus internal state
```

NP Named Pipe data structure

The internal data structure for an instance of a pipe. One of these structures is allocated for each open instance of a particular named pipe.

Allocated NP structures are placed on two lists. The first is headed by ActiveNPList, with list pointer np_next linking together all currently active NP structures.

The second list is headed by the NPN structure defined below and is doubly-linked by the np_flink and np_blink pointers. This list is used to iterate over all instances of a particular pipe name.

Field Name	Offset	Length	Type	Description
np_state	+0	1	B	state of pipe
np_refcnt	+1	1	B	SFT reference count for pipe (1 or 2)
np_next	+2	2	W	pointer to next in active list
np_flink	+4	2	W	pointer to next instance of pipe
np_blink	+6	2	W	pointer to previous instance of pipe
np_namkey	+8	2	W	RMP key value for npn structure
np_scnt	+a	1	B	count of servers (max. 1)
np_ccnt	+b	1	B	count of clients (max. 1)
np_selector1	+c	2	W	selector for outgoing data buffer
np_selector2	+e	2	W	selector for incoming data buffer
np_pipmod	+10	2	W	pipe mode specified at creation time
np_flags	+12	2	W	pipe flags
np_ssft	+14	4	D	back pointer to server SFT
np_csft	+18	4	D	back pointer to client SFT
np_timeo	+1c	4	D	default timeout for DosWaitNmPipe
np_ssem	+20	4	D	server end system semaphore
np_ssemkey	+24	2	W	server's semaphore key
np_csem	+26	4	D	client end system semaphore
np_csemkey	+2a	2	W	client's semaphore key

NPN Named Pipe Name data structure

The following structure contains the common name for the multiple instances of a pipe. Its key value is used as the ProcBlock key for waiters on the pipe. The key value is also used as an RMP key to look up the name record from the NP structure.

Field Name	Offset	Length	Type	Description
npn_link	+0	4	D	pointer to first instance
npn_key	+4	2	W	unique serial number of name
npn_len	+6	2	W	total length of structure
npn_name	+8	254	A	name of pipe, null terminated

NPB Named Pipe Buffer data structure

The following variables are used to control the access to a pipe buffer and are part of the allocated buffer for the pipe. In the case of a duplex pipe, two independent data buffers are allocated. Only one buffer will be allocated for a simplex pipe.

Field Name	Offset	Length	Type	Description
npb_selector	+0	2	W	selector of buffer
npb_first	+2	2	W	base of buffer
npb_in	+4	2	W	next free byte in buffer
npb_out	+6	2	W	next byte of data in buffer
npb_last	+8	2	W	end+1 of buffer
npb_rdlck	+a	2	W	read lock sem.
npb_wtlck	+c	2	W	write lock sem.
npb_rdsem	+e	2	W	read sync sem.
npb_wtsem	+10	2	W	write sync sem.
npb_rdcnt	+12	1	B	count of readers of buffer
npb_wtcnt	+13	1	B	count of writers to buffer
npb_data	+14	2	W	size of data left in pipe

np_state allowable values for named pipe state

Internally, byte stream mode pipes store only a collection of bytes in the data buffer. Message stream mode pipes have individual messages preceeded by a word which indicates the size of the message.

Named pipes may be in one of several states depending on the actions that have been taken on it by the server end and client end. The following state/action table summarizes the valid state transitions:

```
Current state          Action         Next state

 <none>            server MakeNmPipe   DISCONNECTED
 DISCONNECTED      server connect      LISTENING
 LISTENING         client open         CONNECTED
 CONNECTED         server disconn      DISCONNECTED
 CONNECTED         client close        CLOSING
 CLOSING           server disconn      DISCONNECTED
 CONNECTED         server close        CLOSING
 <any other>       server close        <pipe deallocated>
```

A special internal state LISTEN2 is used when a client open is in progress (since some operations may block). This is treated the same as the LISTENING state except that a new open or wait will not recognize it as an available pipe.

If a server disconnects its end of the pipe, the client end will enter a special state in which any future operations (except close) on the file descriptor associated with the pipe will return an error.

Name	Bit Mask	Description
NP_DISCONNECTED	1	after pipe creation or Disconnect
NP_LISTENING	2	after DosNmPipeConnect
NP_CONNECTED	3	after Client open
NP_CLOSING	4	after Client close
NP_LISTEN2	0x12	internal; client open in progress

np_pipmod, sf_pipmod bit mask values:

Name	Bit Mask	Description
NP_NBLK	0x8000	non-blocking read/write
NP_NBLKR	0x8000	non-blocking read
NP_NBLKW	0x8000	non-blocking write
NP_SERVER	0x4000	set if server end
NP_WMESG	0x0400	write messages
NP_RMESG	0x0100	read as messages
NP_TIMOUT	0x3800	Timeout np_sem_blk and np_sem_wait

3.7.11 Anonymous Pipe Structures for OS/2 Warp V3.0

Pointers

SFT field **sfi_hVPB** contains the selector that maps IOBLOCK structure.

Locations

The pipe IOBLOCK is allocated from the kernel heaps.

VM Owner

pipe (0xffa0).

Format

IOBLOCK Anonymous Pipe data structure

A *pipe* is a connection between (among) file handles (JFNs). Data written to the *write end* of the pipe are made available for reading on the *read end*. The $Pipe system call creates a pipe and returns two file handles, one for the read end and one for the write end. These handles are manipulated in the same way as normal file handles; they may be ′dup′ed and are inherited in the same way. Data are written into a pipe via a *write* system call on the write end of the pipe. Likewise, data are read from the pipe via a *read* call on the read end.

Data that are written to a pipe are captured in a circular buffer. The size of the buffer is specified when the pipe is created; if no size is specified, a default size is used.

The circular buffer is described by an *ioblock*. The ioblock is the buffer′s header; the circular buffer proper follows the ioblock in a heap memory object (mapped by a GDT selector) allocated when the pipe is created. The ioblock contains all of the per-pipe information, such as reader, writer, and reference counts, and also holds the pointers into the circular buffer proper.

The selector that points to the circular buffer is stored in the SFT, at sfi_hVPB.
.*

When the in and out pointers are equal, the circular buffer is empty. When the in pointer trails the out pointer by 1, the buffer is full. Thus, a 512 byte buffer can hold only 511 bytes; one byte is lost so that full and empty conditions can be distinguished. In order for the user to be able to put 512 bytes in a pipe that they created with a size of 512, we allow for this byte loss when allocating the segment.

Field Name	Offset	Length	Type	Description
io_inprogcnt	+ 0	1	B	count of read/wrts in progress
io_refcnt	+ 1	1	B	count of references
io_rdrcnt	+ 2	1	B	count of readers
io_wtrcnt	+ 3	1	B	count of writers
io_selector	+ 4	2	W	buffer selector
io_first	+ 6	2	W	ptr to base of circular buffer

Field Name	Offset	Length	Type	Description
io_in	+8	2	W	ptr to next free byte
io_out	+a	2	W	ptr to next byte of data
io_last	+c	2	W	ptr to end+1 of buffer
io_rdlksem	+e	2	W	read lock semaphore
io_wtlksem	+10	2	W	write lock semaphore
io_rdsem	+12	2	W	read sync semaphore
io_wtsem	+14	2	W	write sync semaphore

3.8 I/O System Control Block Reference

The following control blocks are described in this section:

- 3.8.2, “Physical Device Driver Header (DEV) for OS/2 Warp V3.0” on page 209
- 3.8.3, “PDD IQR Information Blocks (DIRQ) for OS/2 Warp V3.0” on page 210
- 3.8.4, “Virtual Device Driver Entry Point Structures” on page 211
- 3.8.5, “Device Driver (Strategy 1) Request Packet (REQ) for OS/2 Warp V3.0” on page 213
- 3.8.6, “BIOS Parameter Block (BPB) for OS/2 Warp V3.0” on page 215

An overview of the I/O system control blocks is as follows:

3.8.1 I/O System Control Block Diagrams

The following diagrams illustrate the relationships between various I/O system control blocks:

- 3.8.1.1, “Physical Device Driver Communication” on page 206
- 3.8.1.2, “Physical Device Driver IRQ Sharing” on page 207
- 3.8.1.3, “Virtual Device Driver Communication” on page 208

3.8.1.1 Physical Device Driver Communication

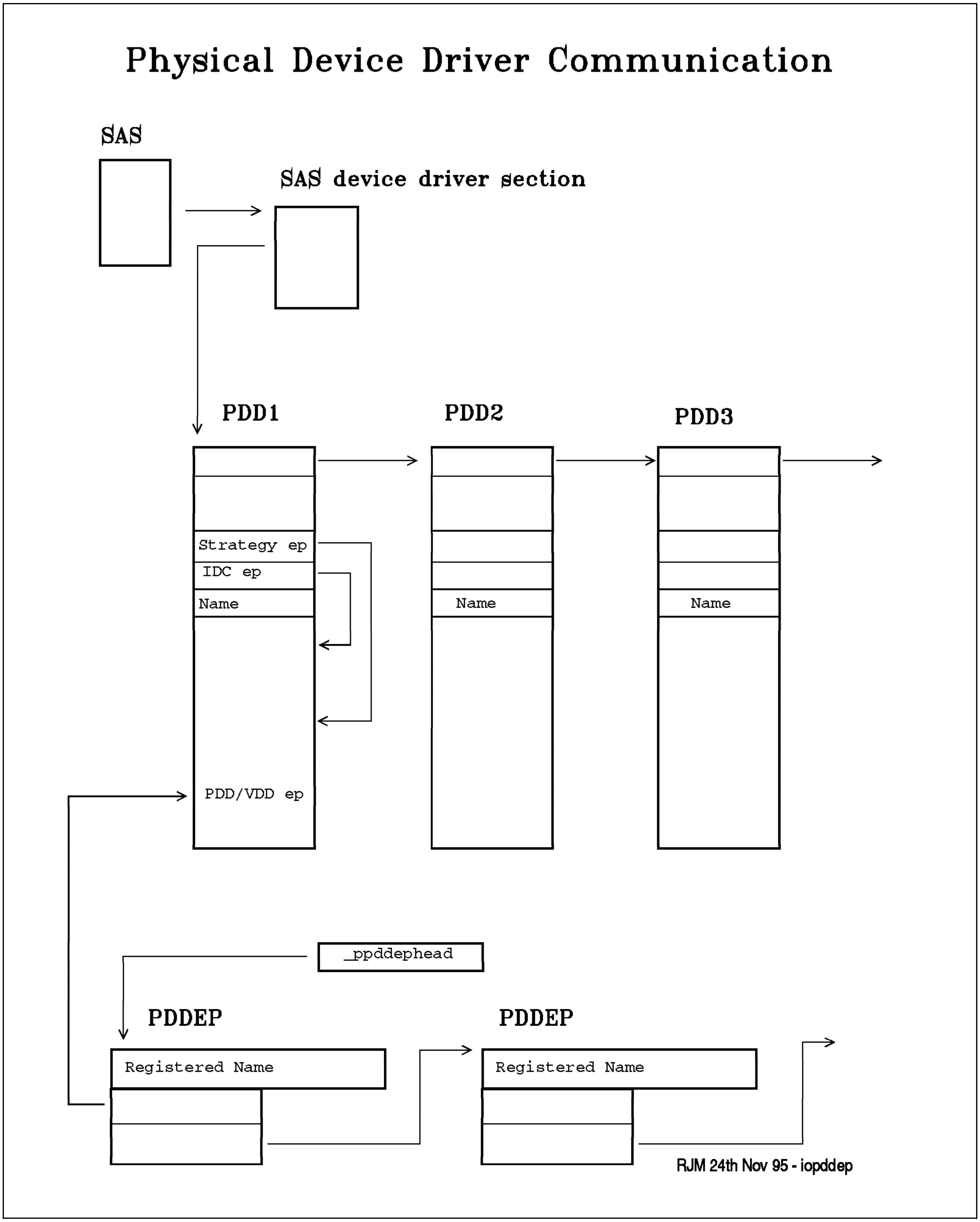


Figure 33. Physical Device Driver Communication

3.8.1.2 Physical Device Driver IRQ Sharing

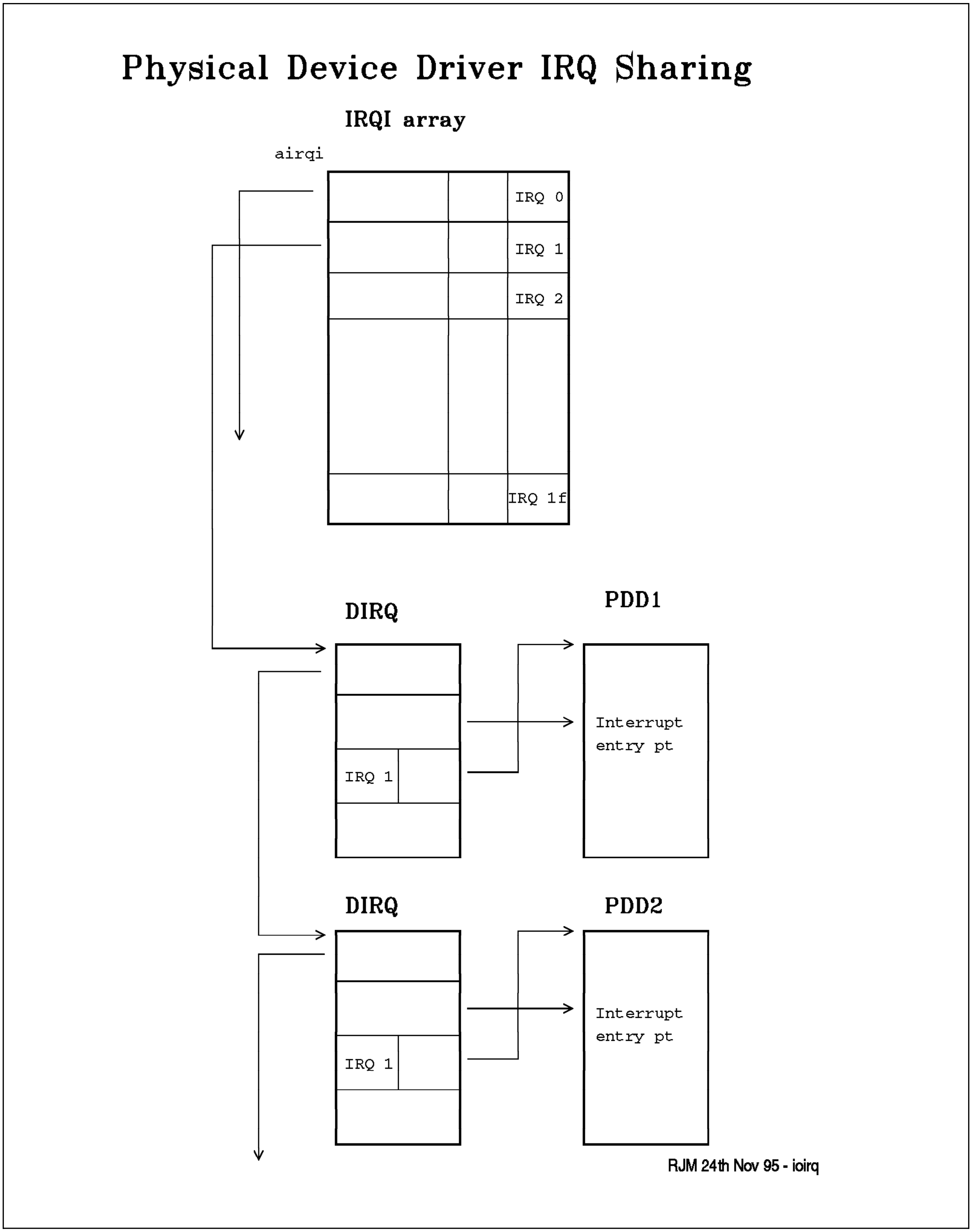


Figure 34. Physical Device Driver IRQ Sharing

3.8.1.3 Virtual Device Driver Communication

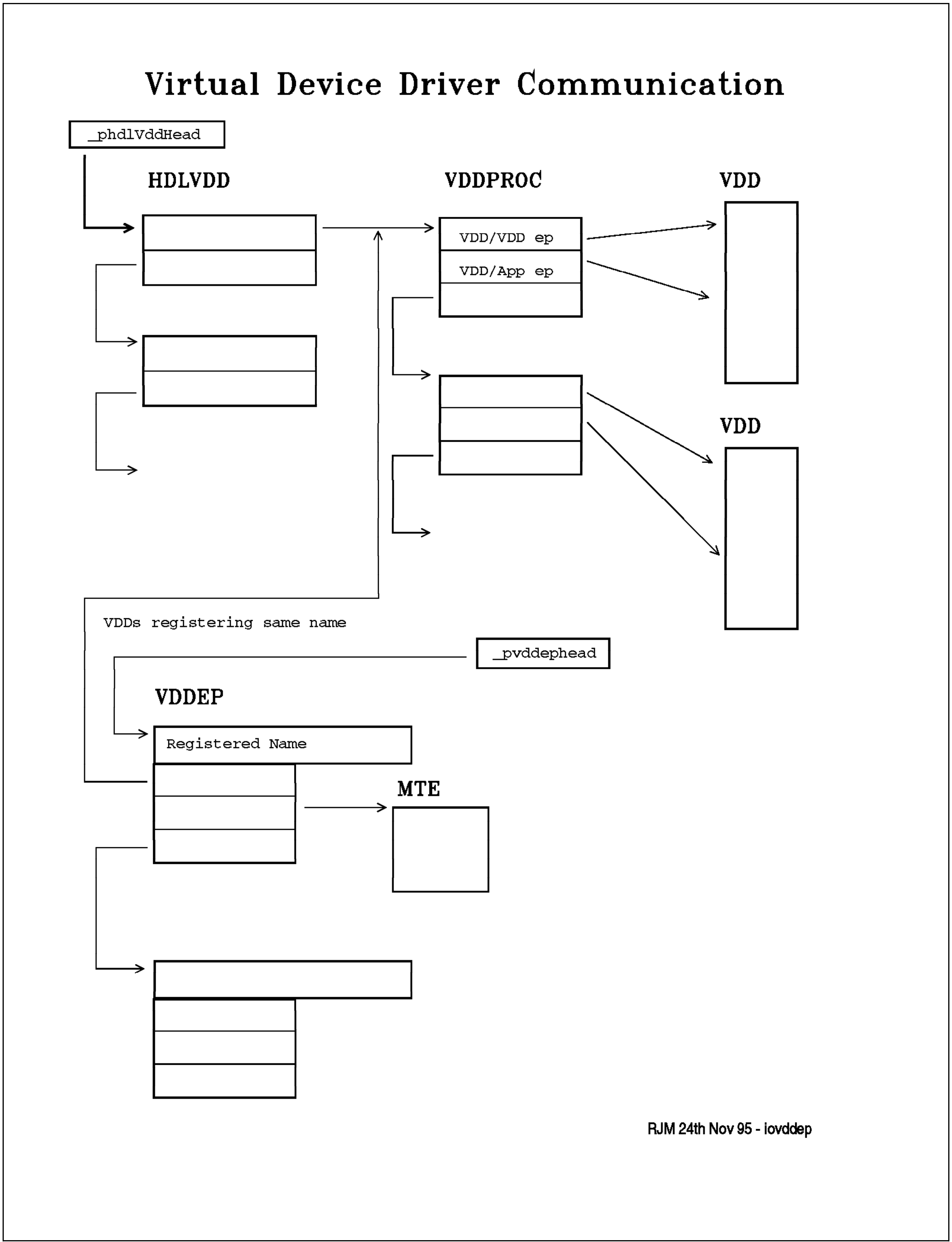

Virtual Device Driver Communication
_phdlVddHead
HDLVDD
VDDPROC
VDD
VDD/VDD ep
VDD/App ep
VDD
VDDs registering same name
_pvddephead
VDDEP
Registered Name
MTE
RJM 24th Nov 95 - iovddep

Figure 35. Virtual Device Driver Communication

3.8.2 Physical Device Driver Header (DEV) for OS/2 Warp V3.0

Pointers

DPB field **dpb_driver_addr** points to the associated Physical Device Driver Header.

SFT field **sf_devptr** points to the associated Physical Device Driver Header.

Locations

Built at the beginning of the first module segment of the device driver.

VM Owner

dd1 (0xff50) to **dd16 (0xff5f)**.

Format

Field Name	Offset	Length	Type	Description
SDevNext	+0	4	D	Pointer to next device header
SDevAtt	+4	2	W	Attributes of the device
SDevStrat	+6	2	W	Strategy entry point
SDevInt	+8	2	W	IDC entry point
SDevName	+a	8	A	name (block uses only 1st byte)
SDevProtCS	+12	2	W	Protect-mode CS of strategy entry pt
SDevProtDS	+14	2	W	Protect-mode DS
SDevRealCS	+16	2	W	Real-mode CS of strategy entry pt
SDevRealDS	+18	2	W	Real-mode DS
SDevCaps	+20	4	D	bit map of DD /MM restrictions

Table 149. SDevCaps Flag Definitions

Name	Bit Mask	Description
DEV_IOCTL2	0x0001	DD can handle dev ioctl2
DEV_16MB	0x0002	DD can handle phys.addresses >16MB
DEV_PARALLEL	0x0004	DD handles parallel port
DEV_ADAPTER_DD	0x0008	DD supports Adapter Dev Driver Intf
DEV_INITCOMPLETE	0x0010	DD can handle CMDInitComplete

Table 150. Device Driver Type defininitions

Name	Bit Mask	Description
DEV_CIN	0x0001	0 2 5 Device is console in
DEV_COUT	0x0002	1 2 5 Device is console out
DEV_NULL	0x0004	2 2 5 Device is the Null device
DEV_CLOCK	0x0008	3 2 5 Device is the clock device
DEV_SPEC	0x0010	4 2 Devices can support INT 29h
DEV_ADD_ON	0x0020	5 Device is add-on driver (BWS)
DEV_GIOCTL	0x0040	6 3 Device supports generic ioctl
DEV_FCNLEV	0x0380	9-7 5 Device function level
DEV_30	0x0800	11 2 5 Accepts Open/Close/Removable Media
DEV_SHARE	0x1000	12 Device wants FS sharing checking
DEV_NON_IBM	0x2000	13 2 5 Device is a non IBM device.
DEV_IOCTL	0x4000	14 2 Device accepts IOCTL request
DEV_CHAR_DEV	0x8000	15 2 5 Device is a character device

Table 151. Level Definitions for Devices

Name	Bit Mask	Description
DEVLEV_0	0x0000	DOS 3.0 and before
DEVLEV_1	0x0080	DOS 5.0
DEVLEV_2	0x0100	OS/2 V1.2 (new gen ioctl iface)
DEVLEV_3	0x0180	OS/2 V2.0 (support of memory above 16MB)

3.8.3 PDD IQR Information Blocks (DIRQ) for OS/2 Warp V3.0

Pointers

IRQI field **irqi_pdirqHead** points to the head of a chain of associated DIRQs.

Locations

airqi locates the table of IRQI entries.

DIRQs are allocated dynamically from the kernel resident heap.

The IRQI array is a static part of the OS2KRNL load module.

VM Owner

DIRQI owner id: **intdirq (0xff78)**.

Format

Table 152. IRQI Table

Field Name	Offset	Length	Type	Description
irqi_pdirqHead	+ 0	4	D	Head of shared DD chain (0 = not set)
irqi_usIRQNum	+ 4	2	W	IRQ number
irqi_usFlags	+ 6	2	W	IRQ Flags

Table 153. irqi_usFlags Flag Definitions

Name	Bit Mask	Description
	0x0003	reserved
irqf_fVDM	0x0004	If set, this IRQ is a candidate for routing to a VDM, if it is not claimed by a PDD
irqf_fNPX	0x0008	If set, the IRQ is the NPX interrupt level
irqf_fSharing	0x0010	If set, the IRQ is sharable. If clear the IRQ can not be shared by DD.
irqf_fSys	0x0020	If set, the IRQ is owned by the system and the handler can not be changed or removed by a device driver. Set initially for the slave, IRQ 2.
irqf_fShared	0x0040	If set, the IRQ can be shared by more than 1 DD. This bit reflects the shared parameter of the first dh_SetIRQ issued for this level.

Table 154. DIRQ Table

Field Name	Offset	Length	Type	Description
dirq_pdirqLink	+ 0	4	D	Next DIRQ structure in list
dirq_f16pfn	+ 4	4	D	DD’s interrupt handler
dirq_usDS	+ 8	2	W	DD’s data segment
dirq_usIRQNum	+ a	2	W	IRQ number
dirq_pdirqFreeList	+ c	4	D	list of unset DIRQs

3.8.4 Virtual Device Driver Entry Point Structures

Pointers

_pvddepHead points to the head of a chain of VDDEP structres. One is allocated for each VDD that register either or both of a VDD/VDD or VDD/OS2 entry point. There entry points are used respectively when either a VDHRequestVDD/VDHOpenVDD or DosRequestVDD/DosOpenVDD call is made.

VDDEP field **vddep_vddp** points to the associated chain of VDDPROC structures. One is allocated for each VDD that registers entry points under the same name.

_phdlVddHead points to the head of a chain of HDLVDD structures. One is allocated for each open VDD. The handle returned is the address of the associated HDLVDD.

_ppddephead points to the head of a chain of PDDEP structures. One is allocated for each Physical Device Driver that registers an entry point for VDD/PDD communication. The entry point is registered using DevHlp_RegisterPDD, and accessed using VDHRequestPDD.

Locations

VDDEPs, VDDPROCs, HDLVDDs and PDDEPs are allocated dynamically from the kernel resident heap.

VM Owner

VDDEP owner ID: **vddep (0xffd2)**.

VDDPROC owner ID: **vddproc (0xffdb)**.

HDLVDD owner ID: **vddlr (0xffd7)**.

PDDEP owner ID: **vddpddep (0xffda)**.

Format

Table 155. VDDEP Format

Field Name	Offset	Length	Type	Description
vddep_szVDD	+0	9	A	VDD Name
vddep_vddp	+9	4	D	VDD entry points (pointer to VDDPROC)
vddep_hmte	+d	4	D	VDD hmte for deregistering if VDD fails
vddep_pvddep	+11	4	D	Next VDD (pointer to next VDDEP)

Table 156. VDDPROC Table

Field Name	Offset	Length	Type	Description
vddproc_pfnvdd	+0	4	D	Entry point for VDD/VDD comm.
vddproc_pfnos2	+4	4	D	Entry point for OS2/VDD comm.
pvddproc	+4	4	D	Entry points registered with same name

Table 157. HDLVDD Table

Field Name	Offset	Length	Type	Description
hdlvdd_pvddproc	+ 0	4	D	VDD routine to be called (pointer to VDDPROC)
hdlvdd	+ 4	4	D	Pointer to next VDD handle; NULL if no more

Table 158. PDDEP Table

Field Name	Offset	Length	Type	Description
pddep_szPDD	+ 0	9	A	PDD name
pddep_fpfn	+ 9	4	D	Entry point routine
pddep_ppddep	+ d	4	D	Next entry point (PDDEP)

3.8.5 Device Driver (Strategy 1) Request Packet (REQ) for OS/2 Warp V3.0

Pointers
TCB field **TCBReqPkt** points to the Request Packet pre-allocated to a thread.

Locations
Allocated from the Request Packet Pool in the System Arena.

VM Owner
reqpkt1 (0xff40).

Format

Field Name	Offset	Length	Type	Description
Packet	+ 0	20	S	Device Driver Request Packet
PktLen	+ 0	1	B	length in bytes of packet
PktUnit	+ 1	1	B	subunit number of block device
PktCmd	+ 2	1	B	command code
PktStatus	+ 3	2	W	status word
PktFlag	+ 5	1	B	disk driver internal flags
	+ 6	3	B	reserved
PktDOSLink	+ 5	4	D	
PktDevLink	+ 9	4	D	device multiple-request link
PktData	+ d	18	S	data pertaining to specific packet
	+ d	10	S	Generic IOCTL

Field Name	Offset	Length	Type	Description
GIOCategory	+d	1	B	Category Code
GIOFunction	+e	1	B	Function code
GIOParaPack	+f	4	D	pointer to parameter packet
GIODataPack	+13	4	D	pointer to data packet
GIOSFN	+17	2	W	(used by Spooler?)
GIOParaLen	+19	2	w	length of parameter packet
GIODataLen	+1b	2	W	length of data packet
	+d	c	S	INIT Command for Base DDs (0 and 27)
InitcUnit	+d	1	B	number of units returned
InitpEnd	+e	4	D	pointer to free mem after dev
InitDevHlp	+e	4	D	address of Device Helper router
InitEcode	+e	2	W	size of code segment
InitEdata	+10	2	W	size of data segment
InitParms	+12	4	D	pointer parameters
InitpBPB	+12	4	D	pointer to BPBs
Initdrv	+16	1	B	drive no. assigned to unit 0
	+17	1	B	reserved
InitSysiData (for resident drivers only)	+18	1	B	SysInit′s DOSALIAS selector
	+d	d	S	query for extended capability command (0x1d)
	+d	3	B	reserved
DCS_Addr	+10	4	W	16:16 of driver caps struc
VCS_Addr	+14	4	W	16:16 of volume char struc
	+d	6	B	Media Check command 1
MedChkmedia	+d	1	B	last media byte seen
MedChkflaga	+e	1	B	-1=change 0=dont know 1=no change
MedChkpVIDa	+f	4	D	pointer to VID
	+d	9	S	build BPB command 2
BldBPBmedia	+d	1	B	media byte
BldBPBbuffer	+e	4	D	scratch buffer
	+d	f	S	Read/Write IO commands 3, 4, 8, 9, 12, 24, 25, 26
IOmedia	+d	1	B	media byte
IOpData	+e	4	D	transfer address
IOcount	+12	2	W	count of bytes/sectors
IOstart	+14	2	W	starting sector (block)
IOPhysRBA	+14	4	D	physical starting sector
IOSFNsRBA	+18	2	W	for device only

Field Name	Offset	Length	Type	Description
PktAdvise	+1a	2	W	for >= v12 only
	+d	4	S	Device Open/Close commands 13 and 14
OCSFN	+d	2	W	sfn of open instance for virtualization
	+d	1	S	Start/Stop console commands (98, 99)
CStpSKG	+d	1	B	Screen/Keyboard number
	+d	6	S	De-install driver command 20
DINEndLocn	+d	4	D	
DINLengthn	+11	2	W	

Table 159. PktStatus Word Masks

Name	Bit Mask	Description
STERR	0x8000	Bit 15 - Error
STINTER	0x0400	Bit 10 - Interim character
STBUI	0x0200	Bit 9 - Busy
STDON	0x0100	Bit 8 - Done
STECODE	0x00ff	Error code
WRECODE	0	

Table 160. PktFlag Flags

Name	Flag value	Description
fPktInt13RP	0x01	Int 13 Request Packet
fPktCallOutDone	0x02	Int 13 Callout completed
fPktDiskIOTchd	0x04	Disk_IO has touched this packet
STDON	0x0100	Bit 8 - Done
STECODE	0x00ff	Error code
WRECODE	0	

3.8.6 BIOS Parameter Block (BPB) for OS/2 Warp V3.0

Pointers

I/O Request Packet fields **InitpBPB** and **BldBPBbuffer** point to the BPB structure.

Locations

Allocated from the System Arena.

VM Owner

Non-specific.

Format

Table 161. BPB Format

Field Name	Offset	Length	Type	Description
BPSECSZ	0	2	W	Size in bytes of physical sector
BPCLUS	2	1	B	Sectors/Alloc unit
BPRES	3	2	W	Number of reserved sectors
BPFTCNT	5	1	B	Number of FATs
BPRDCNT	6	2	W	Number of directory entries
BPSCCNT	8	2	W	Total number of sectors
BPMEDIA	a	1	B	Media descriptor byte
BPFTSEC	b	2	W	Number of sectors taken up by one FAT
BPBSecPerTrack	d	2	W	sectors per track
BPBcHeads	f	2	W	number of heads
BPBcSecHidden	11	2	W	number of hidden sectors before the reserved sectors
BPBcSecHiddenH	13	2	W	High word of hidden sectors
BPBcSecTotal	15	4	D	Big total sectors (if BPSCCNT = 0)
PHYDRV	19	1	B	PHYSICAL DRIVE NUMBER (0 OR 80H)
CURHD	1a	1	B	Unitialized

Chapter 4. Reference Tables

The following reference information is tabulated in this section:

4.1 System Error Codes

OS/2 System Error Codes

Table 162 (Page 1 of 19). OS/2 System Error Codes

Code	Description
0	NO ERROR
1	INVALID FUNCTION
2	FILE NOT FOUND
3	PATH NOT FOUND
4	TOO MANY OPEN FILES
5	ACCESS DENIED
6	INVALID HANDLE
7	ARENA TRASHED
8	NOT ENOUGH MEMORY
9	INVALID BLOCK
10	BAD ENVIRONMENT
11	BAD FORMAT
12	INVALID ACCESS
13	INVALID DATA
15	INVALID DRIVE
16	CURRENT DIRECTORY
17	NOT SAME DEVICE
18	NO MORE FILES
19	WRITE PROTECT
20	BAD UNIT
21	NOT READY

Table 162 (Page 2 of 19). OS/2 System Error Codes

Code	**Description**
22	BAD COMMAND
23	CRC
24	BAD LENGTH
25	SEEK
26	NOT DOS DISK
27	SECTOR NOT FOUND
28	OUT OF PAPER
29	WRITE FAULT
30	READ FAULT
31	GEN FAILURE
32	SHARING VIOLATION
33	LOCK VIOLATION
34	WRONG DISK
35	FCB UNAVAILABLE
36	SHARING BUFFER EXCEEDED
37	CODE PAGE MISMATCHED
38	HANDLE EOF
39	HANDLE DISK FULL
40	BAD COMMAND
41	CRC
42	BAD LENGTH
43	SEEK
44	NOT DOS DISK
45	SECTOR NOT FOUND
46	OUT OF PAPER
47	WRITE FAULT
48	READ FAULT
49	GEN FAILURE
50	NOT SUPPORTED
51	REM NOT LIST
52	DUP NAME
53	BAD NETPATH
54	NETWORK BUSY
55	DEV NOT EXIST
56	TOO MANY CMDS
57	ADAP HDW ERR
58	BAD NET RESP
59	UNEXP NET ERR
60	BAD REM ADAP
61	PRINTQ FULL

Table 162 (Page 3 of 19). OS/2 System Error Codes

Code	**Description**
62	NO SPOOL SPACE
63	PRINT CANCELLED
64	NETNAME DELETED
65	NETWORK ACCESS DENIED
66	BAD DEV TYPE
67	BAD NET NAME
68	TOO MANY NAMES
69	TOO MANY SESS
70	SHARING PAUSED
71	REQ NOT ACCEP
72	REDIR PAUSED
73	SBCS ATT WRITE PROT
74	SBCS GENERAL FAILURE
75	XGA OUT MEMORY
80	FILE EXISTS
81	DUP FCB
82	CANNOT MAKE
83	FAIL I24
84	OUT OF STRUCTURES
85	ALREADY ASSIGNED
86	INVALID PASSWORD
87	INVALID PARAMETER
88	NET WRITE FAULT
89	NO PROC SLOTS
90	NOT FROZEN
90	SYS COMP NOT LOADED
91	ERR TSTOVFL
92	ERR TSTDUP
93	NO ITEMS
95	INTERRUPT
96	INVALID DTA
99	DEVICE IN USE
100	TOO MANY SEMAPHORES
101	EXCL SEM ALREADY OWNED
102	SEM IS SET
103	TOO MANY SEM REQUESTS
104	INVALID AT INTERRUPT TIME
105	SEM OWNER DIED
106	SEM USER LIMIT
107	DISK CHANGE

Table 162 (Page 4 of 19). OS/2 System Error Codes

Code	**Description**
108	DRIVE LOCKED
109	BROKEN PIPE
110	OPEN FAILED
111	BUFFER OVERFLOW
112	DISK FULL
113	NO MORE SEARCH HANDLES
114	INVALID TARGET HANDLE
115	PROTECTION VIOLATION
116	VIOKBD REQUEST
117	INVALID CATEGORY
118	INVALID VERIFY SWITCH
119	BAD DRIVER LEVEL
120	CALL NOT IMPLEMENTED
121	SEM TIMEOUT
122	INSUFFICIENT BUFFER
123	INVALID NAME
123	HPFS INVALID VOLUME CHAR
124	INVALID LEVEL
125	NO VOLUME LABEL
126	MOD NOT FOUND
127	PROC NOT FOUND
128	WAIT NO CHILDREN
129	CHILD NOT COMPLETE
130	DIRECT ACCESS HANDLE
131	NEGATIVE SEEK
132	SEEK ON DEVICE
133	IS JOIN TARGET
134	IS JOINED
135	IS SUBSTED
136	NOT JOINED
137	NOT SUBSTED
138	JOIN TO JOIN
139	SUBST TO SUBST
140	JOIN TO SUBST
141	SUBST TO JOIN
142	BUSY DRIVE
143	SAME DRIVE
144	DIR NOT ROOT
145	DIR NOT EMPTY
146	IS SUBST PATH

Table 162 (Page 5 of 19). OS/2 System Error Codes

Code	**Description**
147	IS JOIN PATH
148	PATH BUSY
149	IS SUBST TARGET
150	SYSTEM TRACE
151	INVALID EVENT COUNT
152	TOO MANY MUXWAITERS
153	INVALID LIST FORMAT
154	LABEL TOO LONG
154	HPFS VOL LABEL LONG
155	TOO MANY TCBS
156	SIGNAL REFUSED
157	DISCARDED
158	NOT LOCKED
159	BAD THREADID ADDR
160	BAD ARGUMENTS
161	BAD PATHNAME
162	SIGNAL PENDING
163	UNCERTAIN MEDIA
164	MAX THRDS REACHED
165	MONITORS NOT SUPPORTED
166	UNC DRIVER NOT INSTALLED
167	LOCK FAILED
168	SWAPIO FAILED
169	SWAPIN FAILED
170	BUSY
171	INT TOO LONG
173	CANCEL VIOLATION
174	ATOMIC LOCK NOT SUPPORTED
175	READ LOCKS NOT SUPPORTED
180	INVALID SEGMENT NUMBER
181	INVALID CALLGATE
182	INVALID ORDINAL
183	ALREADY EXISTS
184	NO CHILD PROCESS
185	CHILD ALIVE NOWAIT
186	INVALID FLAG NUMBER
187	SEM NOT FOUND
188	INVALID STARTING CODESEG
189	INVALID STACKSEG
190	INVALID MODULETYPE

Table 162 (Page 6 of 19). OS/2 System Error Codes

Code	Description
191	INVALID EXE SIGNATURE
192	EXE MARKED INVALID
193	BAD EXE FORMAT
194	ITERATED DATA EXCEEDS 64K
195	INVALID MINALLOCSIZE
196	DYNLINK FROM INVALID RING
197	IOPL NOT ENABLED
198	INVALID SEGDPL
199	AUTODATASEG EXCEEDS 64K
200	RING2SEG MUST BE MOVABLE
201	RELOC CHAIN XEEDS SEGLIM
202	INFLOOP IN RELOC CHAIN
203	ENVVAR NOT FOUND
204	NOT CURRENT CTRY
205	NO SIGNAL SENT
206	FILENAME EXCED RANGE
207	RING2 STACK IN USE
208	META EXPANSION TOO LONG
209	INVALID SIGNAL NUMBER
210	THREAD 1 INACTIVE
211	INFO NOT AVAIL
212	LOCKED
213	BAD DYNALINK
214	TOO MANY MODULES
215	NESTING NOT ALLOWED
216	CANNOT SHRINK
217	ZOMBIE PROCESS
218	STACK IN HIGH MEMORY
219	INVALID EXITROUTINE RING
220	GETBUF FAILED
221	FLUSHBUF FAILED
222	TRANSFER TOO LONG
223	FORCENOSWAP FAILED
224	SMG NO TARGET WINDOW
228	NO CHILDREN
229	INVALID SCREEN GROUP
230	BAD PIPE
231	PIPE BUSY
232	NO DATA
233	PIPE NOT CONNECTED

Table 162 (Page 7 of 19). OS/2 System Error Codes

Code	**Description**
234	MORE DATA
240	VC DISCONNECTED
250	CIRCULARITY REQUESTED
251	DIRECTORY IN CDS
252	INVALID FSD NAME
253	INVALID PATH
254	INVALID EA NAME
255	EA LIST INCONSISTENT
256	EA LIST TOO LONG
257	NO META MATCH
258	FINDNOTIFY TIMEOUT
259	NO MORE ITEMS
260	SEARCH STRUC REUSED
261	CHAR NOT FOUND
262	TOO MUCH STACK
263	INVALID ATTR
264	INVALID STARTING RING
265	INVALID DLL INIT RING
266	CANNOT COPY
267	DIRECTORY
268	OPLOCKED FILE
269	OPLOCK THREAD EXISTS
270	VOLUME CHANGED
271	FINDNOTIFY HANDLE IN USE
272	FINDNOTIFY HANDLE CLOSED
273	NOTIFY OBJECT REMOVED
274	ALREADY SHUTDOWN
275	EAS DIDNT FIT
276	EA FILE CORRUPT
277	EA TABLE FULL
278	INVALID EA HANDLE
279	NO CLUSTER
280	CREATE EA FILE
281	CANNOT OPEN EA FILE
282	EAS NOT SUPPORTED
283	NEED EAS FOUND
284	DUPLICATE HANDLE
285	DUPLICATE NAME
286	EMPTY MUXWAIT
287	MUTEX OWNED

Table 162 (Page 8 of 19). OS/2 System Error Codes

Code	Description
288	NOT OWNER
289	PARAM TOO SMALL
290	TOO MANY HANDLES
291	TOO MANY OPENS
292	WRONG TYPE
293	UNUSED CODE
294	THREAD NOT TERMINATED
295	INIT ROUTINE FAILED
296	MODULE IN USE
297	NOT ENOUGH WATCHPOINTS
298	TOO MANY POSTS
299	ALREADY POSTED
300	ALREADY RESET
301	SEM BUSY
303	INVALID PROCID
304	INVALID PDELTA
305	NOT DESCENDANT
306	NOT SESSION MANAGER
307	INVALID PCLASS
308	INVALID SCOPE
309	INVALID THREADID
310	DOSSUB SHRINK
311	DOSSUB NOMEM
312	DOSSUB OVERLAP
313	DOSSUB BADSIZE
314	DOSSUB BADFLAG
315	DOSSUB BADSELECTOR
316	MR MSG TOO LONG
316	MGS MR MSG TOO LONG
317	MR MID NOT FOUND
318	MR UN ACC MSGF
319	MR INV MSGF FORMAT
320	MR INV IVCOUNT
321	MR UN PERFORM
322	TS WAKEUP
323	TS SEMHANDLE
324	TS NOTIMER
326	TS HANDLE
327	TS DATETIME
328	SYS INTERNAL

Table 162 (Page 9 of 19). OS/2 System Error Codes

Code	**Description**
329	QUE CURRENT NAME
330	QUE PROC NOT OWNED
331	QUE PROC OWNED
332	QUE DUPLICATE
333	QUE ELEMENT NOT EXIST
334	QUE NO MEMORY
335	QUE INVALID NAME
336	QUE INVALID PRIORITY
337	QUE INVALID HANDLE
338	QUE LINK NOT FOUND
339	QUE MEMORY ERROR
340	QUE PREV AT END
341	QUE PROC NO ACCESS
342	QUE EMPTY
343	QUE NAME NOT EXIST
344	QUE NOT INITIALIZED
345	QUE UNABLE TO ACCESS
346	QUE UNABLE TO ADD
347	QUE UNABLE TO INIT
349	VIO INVALID MASK
350	VIO PTR
351	VIO APTR
352	VIO RPTR
353	VIO CPTR
354	VIO LPTR
355	VIO MODE
356	VIO WIDTH
357	VIO ATTR
358	VIO ROW
359	VIO COL
360	VIO TOPROW
361	VIO BOTROW
362	VIO RIGHTCOL
363	VIO LEFTCOL
364	SCS CALL
365	SCS VALUE
366	VIO WAIT FLAG
367	VIO UNLOCK
368	SGS NOT SESSION MGR
369	SMG INVALID SGID

Table 162 (Page 10 of 19). OS/2 System Error Codes

Code	**Description**
369	SMG INVALID SESSION ID
370	SMG NOSG
370	SMG NO SESSIONS
371	SMG GRP NOT FOUND
371	SMG SESSION NOT FOUND
372	SMG SET TITLE
373	KBD PARAMETER
374	KBD NO DEVICE
375	KBD INVALID IOWAIT
376	KBD INVALID LENGTH
377	KBD INVALID ECHO MASK
377	KBD INVALID INPUT MASK
378	KBD INVALID INPUT MASK
379	MON INVALID PARMS
380	MON INVALID DEVNAME
381	MON INVALID HANDLE
382	MON BUFFER TOO SMALL
383	MON BUFFER EMPTY
384	MON DATA TOO LARGE
385	MOUSE NO DEVICE
386	MOUSE INV HANDLE
387	MOUSE INV PARMS
388	MOUSE CAN NOT RESET
389	MOUSE DISPLAY PARMS
390	MOUSE INV MODULE
391	MOUSE INV ENTRY PT
392	MOUSE INV MASK
393	NO MOUSE NO DATA
394	NO MOUSE PTR DRAWN
395	INVALID FREQUENCY
396	NLS NO COUNTRY FILE
396	NO COUNTRY SYS
397	NLS OPEN FAILED
397	OPEN COUNTRY SYS
398	NLS NO CTRY CODE
398	NO COUNTRY OR CODEPAGE
399	NLS TABLE TRUNCATED
400	NLS BAD TYPE
401	NLS TYPE NOT FOUND
401	COUNTRY NO TYPE

Table 162 (Page 11 of 19). OS/2 System Error Codes

Code	Description
402	VIO SMG ONLY
403	VIO INVALID ASCIIZ
404	VIO DEREGISTER
405	VIO NO POPUP
406	VIO EXISTING POPUP
407	KBD SMG ONLY
408	KBD INVALID ASCIIZ
409	KBD INVALID MASK
410	KBD REGISTER
411	KBD DEREGISTER
412	MOUSE SMG ONLY
413	MOUSE INVALID ASCIIZ
414	MOUSE INVALID MASK
415	MOUSE REGISTER
416	MOUSE DEREGISTER
417	SMG BAD ACTION
418	SMG INVALID CALL
419	SCS SG NOTFOUND
420	SCS NOT SHELL
421	VIO INVALID PARMS
422	VIO FUNCTION OWNED
423	VIO RETURN
424	SCS INVALID FUNCTION
425	SCS NOT SESSION MGR
426	VIO REGISTER
427	VIO NO MODE THREAD
428	VIO NO SAVE RESTORE THD
429	VIO IN BG
430	VIO ILLEGAL DURING POPUP
431	SMG NOT BASESHELL
432	SMG BAD STATUSREQ
433	QUE INVALID WAIT
434	VIO LOCK
435	MOUSE INVALID IOWAIT
436	VIO INVALID HANDLE
437	VIO ILLEGAL DURING LOCK
438	VIO INVALID LENGTH
439	KBD INVALID HANDLE
440	KBD NO MORE HANDLE
441	KBD CANNOT CREATE KCB

Table 162 (Page 12 of 19). OS/2 System Error Codes

Code	**Description**
442	KBD CODEPAGE LOAD INCOMPL
443	KBD INVALID CODEPAGE ID
444	KBD NO CODEPAGE SUPPORT
445	KBD FOCUS REQUIRED
446	KBD FOCUS ALREADY ACTIVE
447	KBD KEYBOARD BUSY
448	KBD INVALID CODEPAGE
449	KBD UNABLE TO FOCUS
450	SMG SESSION NON SELECT
451	SMG SESSION NOT FOREGRND
452	SMG SESSION NOT PARENT
453	SMG INVALID START MODE
454	SMG INVALID RELATED OPT
455	SMG INVALID BOND OPTION
456	SMG INVALID SELECT OPT
457	SMG START IN BACKGROUND
458	SMG INVALID STOP OPTION
459	SMG BAD RESERVE
460	SMG PROCESS NOT PARENT
461	SMG INVALID DATA LENGTH
462	SMG NOT BOUND
463	SMG RETRY SUB ALLOC
464	KBD DETACHED
465	VIO DETACHED
466	MOU DETACHED
467	VIO FONT
468	VIO USER FONT
469	VIO BAD CP
470	VIO NO CP
471	VIO NA CP
472	INVALID CODE PAGE
473	CPLIST TOO SMALL
474	CP NOT MOVED
475	MODE SWITCH INIT
476	CODE PAGE NOT FOUND
477	UNEXPECTED SLOT RETURNED
478	SMG INVALID TRACE OPTION
479	VIO INTERNAL RESOURCE
480	VIO SHELL INIT
481	SMG NO HARD ERRORS

Table 162 (Page 13 of 19). OS/2 System Error Codes

Code	Description
482	CP SWITCH INCOMPLETE
483	VIO TRANSPARENT POPUP
484	CRITSEC OVERFLOW
485	CRITSEC UNDERFLOW
486	VIO BAD RESERVE
487	INVALID ADDRESS
488	ZERO SELECTORS REQUESTED
489	NOT ENOUGH SELECTORS AVA
490	INVALID SELECTOR
491	SMG INVALID PROGRAM TYPE
492	SMG INVALID PGM CONTROL
493	SMG INVALID INHERIT OPT
494	VIO EXTENDED SG
495	VIO NOT PRES MGR SG
496	VIO SHIELD OWNED
497	VIO NO MORE HANDLES
498	VIO SEE LOG
499	VIO ASSOCIATED DC
500	KBD NO CONSOLE
501	MOUSE NO CONSOLE
502	MOUSE INVALID HANDLE
503	SMG INVALID DEBUG PARMS
504	KBD EXTENDED SG
505	MOU EXTENDED SG
506	SMG INVALID ICON FILE
507	TRC PID NON EXISTENT
508	TRC COUNT ACTIVE
509	TRC SUSPENDED BY COUNT
510	TRC COUNT INACTIVE
511	TRC COUNT REACHED
512	NO MC TRACE
513	MC TRACE
514	TRC COUNT ZERO
515	SMG TOO MANY DDS
516	SMG INVALID NOTIFICATION
517	LF INVALID FUNCTION
518	LF NOT AVAIL
519	LF SUSPENDED
520	LF BUF TOO SMALL
521	LF BUFFER CORRUPTED

Table 162 (Page 14 of 19). OS/2 System Error Codes

Code	**Description**
521	LF BUFFER FULL
522	LF INVALID DAEMON
522	LF INVALID RECORD
523	LF INVALID TEMPL
523	LF INVALID SERVICE
524	LF GENERAL FAILURE
525	LF INVALID ID
526	LF INVALID HANDLE
527	LF NO ID AVAIL
528	LF TEMPLATE AREA FULL
529	LF ID IN USE
530	MOU NOT INITIALIZED
531	MOUINITREAL DONE
532	DOSSUB CORRUPTED
533	MOUSE CALLER NOT SUBSYS
534	ARITHMETIC OVERFLOW
535	TMR NO DEVICE
536	TMR INVALID TIME
537	PVW INVALID ENTITY
538	PVW INVALID ENTITY TYPE
539	PVW INVALID SPEC
540	PVW INVALID RANGE TYPE
541	PVW INVALID COUNTER BLK
542	PVW INVALID TEXT BLK
543	PRF NOT INITIALIZED
544	PRF ALREADY INITIALIZED
545	PRF NOT STARTED
546	PRF ALREADY STARTED
547	PRF TIMER OUT OF RANGE
548	PRF TIMER RESET
549	HPFS CHKDSK NO PARM SPACE
550	HPFS CHKDSK NORECOGNIZE
551	HPFS CHKDSK NOROOT FIND
552	HPFS CHKDSK NOFIX FS ERROR
553	HPFS CHKDSK CORRECT FS ERR
554	HPFS CHKDSK ORGAN FIX
555	HPFS CHKDSK RELOC BBPDATA
556	HPFS CHKDSK REM CORRU BLOC
557	HPFS CHKDSK REM CORRUP FIL
558	HPFS CHKDSK FIX SPACE ALLO

Table 162 (Page 15 of 19). OS/2 System Error Codes

Code	Description
559	HPFS NOT FORMATTED DISK
560	HPFS CHKDSK COR ALLOC
561	HPFS CHKDSK SEARC UNALLOC
562	HPFS CHKDSK DET LOST DATA
563	HPFS CHKDSK PERCENT SEARC
564	HPFS CHKDSK LOST DATASEARC
565	HPFS CHKDSK CRIT NOREAD
566	HPFS CHKDSK DISK INUSE
567	HPFS CHKDSK RECOVTEMP RELOC
568	HPFS TOTAL DISK SPACE
569	HPFS DIR KBYTES
570	HPFS FILE KBYTES
571	HPFS KBYTES AVAILABLE
572	HPFS CHKDSK PLACE REC FILE
573	HPFS CHKDSK RECO DIR AS
574	HPFS CHKDSK PLACEED DATA
575	HPFS CHKDSK RECOV EA
576	HPFS CHKDSK FIND EA INTEM
577	HPFS CHKDSK RELOC TEMP EA
578	HPFS CHKDSK RELOC AC LIST
579	HPFS CHKDSK LIST NORELOC
580	HPFS CHKDSK TRUN EA LIST
581	HPFS CHKDSK TRUN EA NAME
582	HPFS CHKDSK TRUN EA BBLOCK
583	HPFS CHKDSK REM INVALID EA
584	HPFS CHKDSK FIX EA ALLOC
585	HPFS CHKDSK FIX ALACCCTRL
586	HPFS CHKDSK ACCTR LIST BBL
587	HPFS CHKDSK REM ACLIST
588	HPFS CHKDSK FOUND DATANORL
589	HPFS WRONG VERSION
590	HPFS CHKDSK FOUND DATATEMP
591	HPFS CHKDSK FIX TEMPSTATUS
592	HPFS CHKDSK FIX NEEDEADATA
593	HPFS RECOVER PARM ERROR
594	HPFS RECOV FILE NOT FOUND
595	HPFS RECOV UNKNOWN ERROR
596	HPFS RECOV NOT ENOUGH MEM
597	HPFS RECOV NOWRITE DATA
598	HPFS RECOV NOTEMP CREATE

Table 162 (Page 16 of 19). OS/2 System Error Codes

Code	**Description**
599	HPFS RECOV EA NOREAD
600	HPFS RECOV FILE BYTES
601	HPFS RECOV BAD BYTES RECOV
602	HPFS RECOV FILEBYTES NOREC
603	HPFS RECOV DISK INUSE
604	HPFS RECOV FILE NODELETE
605	HPFS RECOV NOCREATE NEWFILE
606	HPFS RECOV SYSTEM ERROR
607	HPFS SYS PARM ERROR
608	HPFS SYS CANNOT INSTALL
609	HPFS SYS DRIVE NOTFORMATED
610	HPFS SYS FILE NOCREATE
611	HPFS SIZE EXCEED
612	HPFS SYNTAX ERR
613	HPFS NOTENOUGH MEM
614	HPFS WANT MEM
615	HPFS GET RETURNED
616	HPFS SET RETURNED
617	HPFS BOTH RETURNED
618	HPFS STOP RETURNED
619	HPFS SETPRTYRETURNED
620	HPFS ALCSG RETURNED
621	HPFS MSEC SET
622	HPFS OPTIONS
623	HPFS POS NUM VALUE
624	HPFS VALUE TOO LARGE
625	HPFS LAZY NOT VALID
626	HPFS VOLUME ERROR
627	HPFS VOLUME DIRTY
628	HPFS NEW SECTOR
629	HPFS FORMAT PARM ERROR
630	HPFS CANNOT ACCESS CONFIG
631	HPFS RECOV FILE
632	HPFS CHKDSK KBYTES RESERVE
633	HPFS CHKDSK KBYTES IN EA
634	HPFS BYTEBUF SET
635	HPFS FORMATTING COMPLETE
636	HPFS WRONG VOLUME LABEL
637	HPFS FMAT TOO MANY DRS
638	VDD UNSUPPORTED ACCESS

Table 162 (Page 17 of 19). OS/2 System Error Codes

Code	**Description**
639	VDD LOCK USEAGE DENIED
640	TIMEOUT
641	VDM DOWN
642	VDM LIMIT
643	VDD NOT FOUND
644	INVALID CALLER
645	PID MISMATCH
646	INVALID VDD HANDLE
647	VLPT NO SPOOLER
648	VCOM DEVICE BUSY
649	VLPT DEVICE BUSY
650	NESTING TOO DEEP
651	VDD MISSING
671	BIDI INVALID LENGTH
672	BIDI INVALID INCREMENT
673	BIDI INVALID COMBINATION
674	BIDI INVALID RESERVED
675	BIDI INVALID EFFECT
676	BIDI INVALID CSDREC
677	BIDI INVALID CSDSTATE
678	BIDI INVALID LEVEL
679	BIDI INVALID TYPE SUPPORT
680	BIDI INVALID ORIENTATION
681	BIDI INVALID NUM SHAPE
682	BIDI INVALID CSD
683	BIDI NO SUPPORT
684	NO BIDI RW INCOMPLETE
689	HPFS LAZY ON
690	HPFS LAZY OFF
691	IMP INVALID PARM
692	IMP INVALID LENGTH
693	MSG HPFS DISK WARN
694	MSG HPFS FNODE WARN
730	MON BAD BUFFER
731	MODULE CORRUPTED
732	BOOT DRIVE NOT ACCESSIBLE
1477	SM OUTOF SWAPFILE
2055	LF TIMEOUT
2057	LF SUSPEND SUCCESS
2058	LF RESUME SUCCESS

Table 162 (Page 18 of 19). OS/2 System Error Codes

Code	Description
2059	LF REDIRECT SUCCESS
2060	LF REDIRECT FAILURE
32768	SWAPPER NOT ACTIVE
32769	INVALID SWAPID
32770	IOERR SWAP FILE
32771	SWAP TABLE FULL
32772	SWAP FILE FULL
32773	CANT INIT SWAPPER
32774	SWAPPER ALREADY INIT
32775	PMM INSUFFICIENT MEMORY
32776	PMM INVALID FLAGS
32777	PMM INVALID ADDRESS
32778	PMM LOCK FAILED
32779	PMM UNLOCK FAILED
32780	PMM MOVE INCOMPLETE
32781	UCOM DRIVE RENAMED
32782	UCOM FILENAME TRUNCATED
32783	UCOM BUFFER LENGTH
32784	MON CHAIN HANDLE
32785	MON NOT REGISTERED
32786	SMG ALREADY TOP
32787	PMM ARENA MODIFIED
32788	SMG PRINTER OPEN
32789	PMM SET FLAGS FAILED
32790	INVALID DOS DD
32791	BLOCKED
32792	NOBLOCK
32793	INSTANCE SHARED
32794	NO OBJECT
32795	PARTIAL ATTACH
32796	INCACHE
32797	SWAP IO PROBLEMS
32798	CROSSES OBJECT BOUNDARY
32799	LONGLOCK
32800	SHORTLOCK
32801	UVIRTLOCK
32802	ALIASLOCK
32803	ALIAS
32804	NO MORE HANDLES
32805	SCAN TERMINATED

Table 162 (Page 19 of 19). OS/2 System Error Codes

Code	**Description**
32806	TERMINATOR NOT FOUND
32807	NOT DIRECT CHILD
32808	DELAY FREE
32809	GUARDPAGE
32900	SWAPERROR
32901	LDRERROR
32902	NOMEMORY
32903	NOACCESS
32904	NO DLL TERM
65026	CPSIO CODE PAGE INVALID
65027	CPSIO NO SPOOLER
65028	CPSIO FONT ID INVALID
65033	CPSIO INTERNAL ERROR
65034	CPSIO INVALID PTR NAME
65037	CPSIO NOT ACTIVE
65039	CPSIO PID FULL
65040	CPSIO PID NOT FOUND
65043	CPSIO READ CTL SEQ
65045	CPSIO READ FNT DEF
65047	CPSIO WRITE ERROR
65048	CPSIO WRITE FULL ERROR
65049	CPSIO WRITE HANDLE BAD
65074	CPSIO SWIT LOAD
65077	CPSIO INV COMMAND
65078	CPSIO NO FONT SWIT
65079	ENTRY IS CALLGATE
0xFF00	USER DEFINED BASE

DOS INT 24 Critical Error Codes

Table 163 (Page 1 of 2). DOS INT 24 Critical Error Codes

Code	**Description**
0	I24 WRITE PROTECT
1	I24 BAD UNIT
2	I24 NOT READY
3	I24 BAD COMMAND
4	I24 CRC
5	I24 BAD LENGTH
6	I24 SEEK
7	I24 NOT DOS DISK

Table 163 (Page 2 of 2). DOS INT 24 Critical Error Codes

Code	Description
8	I24 SECTOR NOT FOUND
9	I24 OUT OF PAPER
10	I24 WRITE FAULT
11	I24 READ FAULT
12	I24 GEN FAILURE
13	I24 DISK CHANGE
15	I24 WRONG DISK
16	I24 UNCERTAIN MEDIA
17	I24 CHAR CALL INTERRUPTED
18	I24 NO MONITOR SUPPORT
19	I24 INVALID PARAMETER
20	I24 DEVICE IN USE
21	I24 QUIET INIT FAIL

4.2 OS/2 System Exception Codes

The exception values are 32-bit values laid out as follows:

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
┌───┬─┬─────────────────────────┬───────────────────────────────┐
│Sev│C│        Facility         │             Code              │
└───┴─┴─────────────────────────┴───────────────────────────────┘

where

    Sev - is the severity code
        00 - Success
        01 - Informational
        10 - Warning
        11 - Error

    C - is the Customer code flag

    Facility - is the facility code

    Code - is the facility′s status code

Exceptions specific to OS/2 2.0 (e.g. XCPT_SIGNAL) will be marked
with a facility code of 1.
```

Figure 36. Exception Values Layout

80000001H
XCPT_GUARD_PAGE_VIOLATION

P1 Access Code

00000001H	XCPT_READ_ACCESS
00000002H	XCPT_WRITE_ACCESS

P2 FaultAddr

80010001H
XCPT_UNABLE_TO_GROW_STACK

0C0010001H
XCPT_PROCESS_TERMINATE

0C0010002H
XCPT_ASYNC_PROCESS_TERMINATE

P1 TID of terminator thread

0C0010003H
XCPT_SIGNAL

P1 Signal Number

1	XCPT_SIGNAL_INTR
3	XCPT_SIGNAL_KILLPROC
4	XCPT_SIGNAL_BREAK

0C0010004H
XCPT_B1NPX_ERRATA_02

0C0000005H
XCPT_ACCESS_VIOLATION

This relates to Traps 0x09, 0x0b, 0x0c, 0x0d and 0x0e.

P1 Access Code

00000000H	XCPT_UNKNOWN_ACCESS
00000001H	XCPT_READ_ACCESS
00000002H	XCPT_WRITE_ACCESS
00000004H	XCPT_EXECUTE_ACCESS
00000008H	XCPT_SPACE_ACCESS
00000010H	XCPT_LIMIT_ACCESS

P2

FaultAddr	XCPT_READ_ACCESS/XCPT_WRITE_ACCESS
Selector	XCPT_SPACE_ACCESS
-1	XCPT_LIMIT_ACCESS

0C0000006H
XCPT_IN_PAGE_ERROR

This relates to Trap 0x0e.

P1 FaultAddr

0C000001CH
XCPT_ILLEGAL_INSTRUCTION

This relates to Trap 0x06.

0C000001DH
XCPT_INVALID_LOCK_SEQUENCE

0C0000024H
XCPT_NONCONTINUABLE_EXCEPTION

0C0000025H
XCPT_INVALID_DISPOSITION

0C0000026H
XCPT_UNWIND

0C0000027H
XCPT_BAD_STACK

0C0000028H
XCPT_INVALID_UNWIND_TARGET

0C0000093H
XCPT_ARRAY_BOUNDS_EXCEEDED

This relates to Trap 0x05.

0C0000094H
XCPT_FLOAT_DENORMAL_OPERAND

This relates to Trap 0x10.

0C0000095H
XCPT_FLOAT_DIVIDE_BY_ZERO

This relates to Trap 0x10.

0C0000096H
XCPT_FLOAT_INEXACT_RESULT

This relates to Trap 0x10.

0C0000097H
XCPT_FLOAT_INVALID_OPERATION

This relates to Trap 0x10.

0C0000098H
XCPT_FLOAT_OVERFLOW

This relates to Trap 0x10.

0C0000099H
XCPT_FLOAT_STACK_CHECK

This relates to Trap 0x10.

0C000009AH
XCPT_FLOAT_UNDERFLOW

This relates to Trap 0x10.

0C000009BH
XCPT_INTEGER_DIVIDE_BY_ZERO

This relates to Trap 0x00.

0C00009CH
XCPT_INTEGER_OVERFLOW

This relates to Trap 0x04.

0C00009DH
XCPT_PRIVILEGED_INSTRUCTION

This relates to Trap 0x0d.

0C00009EH
XCPT_DATATYPE_MISALIGNMENT

This relates to Trap 0x11.

P1	Access Code
	00000001H XCPT_READ_ACCESS **00000002H** XCPT_WRITE_ACCESS
P2	Alignment
P3	FaultAddr

0C00009FH
XCPT_BREAKPOINT

This relates to Trap 0x03.

0C0000A0H
XCPT_SINGLE_STEP

This relates to Trap 0x01.

For further information refer to:

- *OS/2 Technical Library - Control Program Programming Reference,* Appendix C.
- The BSEXCPT.H or BSEXCPT.INC include files supplied with the OS/2 Programmers Toolkit.

4.3 Trap Screen Reference

The trap screen has two basic formats:

The application trap (SYS3175 and SYS3171) messages.

The Internal Processing Error (IPE).

```
01-> 08-09-1995  17:22:41  SYS3171  PID 0054
02-> E:\RJM\INVERTP\INVERTP.EXE
03-> c0000005
04-> 00010267
05-> P1=00000008  P2=6d640000  P3=XXXXXXXX  P4=XXXXXXXX
06-> EAX=00000000  EBX=00000000  ECX=00000000  EDX=00000000
07-> ESI=00000000  EDI=00000000
08-> DS=0053  DSACC=d0f3  DSLIM=1bffffff
09-> ES=0053  ESACC=d0f3  ESLIM=1bffffff
10-> FS=150b  FSACC=00f3  FSLIM=00000030
11-> GS=0000  GSACC=****  GSLIM=********
12-> CS:EIP=005b:00010267  CSACC=d0df  CSLIM=1bffffff
13-> SS:ESP=0000:00201ff0  SSACC=****  SSLIM=********
14-> EBP=00201ff4  FLG=00002306

15-> INVERTP.EXE 0001:00000267
```

Figure 37. Application Trap

The information presented varies slightly according to circumstance. In general, inapplicable information is either omitted or overlayed with asterisks (*) or Xs.

Each line of the trap screen conveys the following meaning:

1. Date and Time or Trap, Trap message ID and Failing Process ID. This may also include the thread slot number and module handle.
2. Failing module.
3. Exception code. See 4.2, “OS/2 System Exception Codes” on page 237 for a complete set of system generated exceptions.
4. Instruction address at time of exception.
5. Exception Information Parameters. See 4.2, “OS/2 System Exception Codes” on page 237 for the exception information parameters that are associated with each system exception.
6. The EAX, EXB, EXC and EDX registers at the time the exception was reported.
7. The ESI and EDI registers at the time the exception was reported.
8. The DS selector at the time the exception was reported.

   This information is presented in the form:

   xS=nnnn The selector value.

xSACC=nnnn The descriptor access bits.

Reading from right to left the bits of the access field are assigned the following meaning:

0	(A) 1=Accessed
1	(W) 1=Writeable
2	(E) 1=Executable
3	0
4	(S) 1=Application 0=System
5 & 6	(DPL) Privilege Level
7	(P) 1=Segment present
8 - 11	0
12	(AVL) 1=UVIRT allocation
13	(D) 1=32-bit Operands/Data
14	0
15	(G) 1=4K granularity limit, 0=byte granularity limit

See the *INTEL Pentium User's Guide,* Volume 3 for more information on descriptor formats.

xSLIM=nnnnnnnn The limit address from the descriptor.

9. The ES selector at the time the exception was reported.
10. The FS selector at the time the exception was reported.
11. The GS selector at the time the exception was reported.
12. The instruction address at the time the exception was reported, followed by the CS selector Limit and Access fields.
13. The stack address at the time the exception was reported, followed by the SS selector Limit and Access fields.
14. The EBP register and EFLAGS register.
15. The module name and relative object and offset within the module that corresponds the the CS:EIP at the time of exception.

System Internal Processing Error (IPE)

The IPE message appears because of a fatal internal error condition. This may or may not be a trap, although the IPE trap is the most common.

The IPE message has the general format:

```
1-> <IPE specific Message>

2->  THE SYSTEM DETECTED AN INTERNAL PROCESSING
     ERROR AT LOCATION ##xxxx:yyyyyyyy - aaaa:bbbb

3->  eeeee , llll
4->  038600d1
5-> INTERNAL REVISION 6 . 307  DATE: 92/03/01
```

The parts of the IPE message are:

1. IPE specific message, which could be a simple line of text, for example:

```
CPS: Empty allocation block--not supported.
```

or a formatted register dump for a system trap, such as:

```
 TRAP 0002        ERRCD= 0000  ERACC= ****  ERLIM= ********
  EAX= 7d240a58  EBX= ff202fdc  ECX= 00064423  EDX= 00003624
  ESI= fff3272c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
  CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
  SS:ESP= 0030 : 00004a38  SSACC= 1097  SSLIM= 00003fff
  DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= fffffffb
  ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
  FS= 0000  FSACC= ****  FSLIM= ********
  GS= 0000  GSACC= ****  GSLIM= ********
```

Figure 38. Formatted Regsister Dump for a System Trap

2. The CS:EIP of the caller to the kernel panic routine is shown as ***##xxxx:yyyyyyyy***. For traps this will always be an address within the trap handler and not the address at which the error occurred, which is given in the error specific message.

   The CS:EIP is prefixed with either ## to indicate protect mode, paging enables in accordance with the Kernel Debugger command prompt.

   The kernel relative object:offset address is shown as ***aaaa:bbbb***.

3. ***eeeee*** is intended to be an internal error code and ***llll*** source line number information. These may not contain meaningful data.
4. The processor ID.
5. The kernel revision information.

An example of the IPE trap screen is show in the following diagram:

```
1->  TRAP 0002        ERRCD= 0000  ERACC= ****  ERLIM= ********
      EAX= 7d240a58  EBX= ff202fdc  ECX= 00064423  EDX= 00003624
      ESI= fff3272c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
2->   CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
      SS:ESP= 0030 : 00004a38  SSACC= 1097  SSLIM= 00003fff
3->   DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= fffffffb
4->   ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
      FS= 0000  FSACC= ****  FSLIM= ********
      GS= 0000  GSACC= ****  GSLIM= ********
     THE SYSTEM DETECTED AN INTERNAL PROCESSING
     ERROR AT LOCATION ##0160:fff6453f - 000d:a53f

     60000 , 9084
     038600d1
     INTERNAL REVISION 6 . 307  DATE: 92/03/01
```

Figure 39. IPE Trap Screen

The register information may be interpreted to be for application trap screens, with the following notes:

1. This line shows the trap number followed by the INTEL error code. Most often the associated error code is a selector number. When this is the case, this line formats the selector′s access and limit values.

2. This line shows the address at which the trap occurred.

3. The value of control register 0 (CR0) is formatter after the DS register.

   CR0 contains processor control mode settings.

4. The value of control register 2 (CR2) is formatter after the ES register.

   CR2 contains the fault address for TRAP E errors.

4.4 Standard GDT Assignments

The following table lists the GDT assignments that are statically assigned or assigned dynamically during initialization.

This list is subject to change from release to release but may be verified by listing symbols from OS2KRNL segment DOSGDTDATA using the Kernel Debugger LS command.

Table 164 (Page 1 of 3). GDT Assignments

Selector	**Symbol**	**Description**
0	GDT	entry 0 is reserved (invalid)
8	GDT_GDT	entry 8 used to be GDT (now invalid)
10	GDT_TSS	Protect mode TSS
18	GDT_IDT	Protect Mode IDT
20	GDT_RM_IDT	Selector for 1st 1K
28	GDT_LDT	Selector for LDT
30	GDT_PTDA	PTDA/TCB/TSD selector
38	GDT_FPEM	Floating Point Emulator Work Area
40	GDT_ROMDATA	ROM data at 40h
4c	GDT_R2DS	Ring 2 Data Selector
53	GDT_R3DS	Ring 3 Data Selector
5b	GDT_R3CS	Ring 3 Code Selector
63	GDT_R3PDS	Ring 3 Protected Data Selector
6b	GDT_R3THKDS,	Ring 3 Thunk Data Selector
70	GDT_SAS,	System Anchor Segment
78	GDT_DOSALIAS	SAS Read/Write Alias
80	GDT_SYSINFOSEG	InfosegGDT
88	GDT_DFTSS	Double Fault TSS
90	GDT_DFSTACK	Trap 8 stack selector
98	GDT_VPB	VPB BMP Segment
a0	GDT_RDR1	Reserved
a8	GDT_Buffers	Buffer Pool Segment
b0	GDT_Unused	unused selector (used to be MFT)
b8	GDT_RLR	RLR selector
c0	GDT_SFT	SFT selector of first SFT segment
c8	GDT_FSC	FSC array segment selector
d0	GDT_mFSD	mini-FSD
d8	GDT_RIPL	Remote IPL data
e0	GDT_NULLIDT	Invalid descriptor for mode switch
e8	GDT_INTSTACK	Interrupt stack alias
f0	GDT_RMCODE	386 modesw code selector
f8	GDT_RMDATA	386 modesw data selector
100	DOSHLP_CODESEL	DosHlp Code Selector

Table 164 (Page 2 of 3). GDT Assignments

Selector	**Symbol**	**Description**
108	GDT_Pool	Start of dynamic GDT allocations
1508	GDT_Poolend	End of dynamic GDT allocations
150b	GDT_TIB	TIB selector
1d10	GDT_DOSALLOCSEG	DOSALLOCSEG call gate
1d18	GDT_DOSALLOCPROTSEG	DOSALLOCPROTSEG call gate
1d20	GDT_DOSDYNAMICTRACE	DOSDYNAMICTRACE call gate
1d28	GDT_DOSERROR	DOSERROR call gate
1d30	GDT_DOSFREERESOURCE	DOSFREERESOURCE call gate
1d38	GDT_DOSQUERYABIOSSUPPORT	DOSQUERYABIOSSUPPORT call gate
1d40	GDT_DOS16LDRDIRTYWORKER	DOS16LDRDIRTYWORKER call gate
1d48	GDT_DOSFREESEG	DOSFREESEG call gate
1d50	GDT_DOSGETPROCADDR	DOSGETPROCADDR call gate
1d58	GDT_DOSIEXECPGM	DOSIEXECPGM call gate
1d60	GDT_DOSIQAPPTYPE	DOSIQAPPTYPE call gate
1d68	GDT_DOSISEMWAIT	DOSISEMWAIT call gate
1d70	GDT_DOSLOADMODULE	DOSLOADMODULE call gate
1d78	GDT_DOSMAKEPIPE	DOSMAKEPIPE call gate
1d80	GDT_DOSREALLOCSEG	DOSREALLOCSEG call gate
1d88	GDT_DOSSICG	DOSSICG call gate
1d90	GDT_PANICWRITE	PANICWRITE call gate
1d98	GDT_DOSSETPRTY	DOSSETPRTY call gate
1da0	GDT_DOSLOGMODE	DOSLOGMODE call gate
1da8	GDT_DOSSETCP	DOSSETCP call gate
1db0	GDT_DOSGLOBALSEG	DOSGLOBALSEG call gate
1db8	GDT_DOSCREATETHREAD	DOSCREATETHREAD call gate
1dc0	GDT_DOSEXIT	DOSEXIT call gate
1dc8	GDT_DOSEXITLIST	DOSEXITLIST call gate
1dd0	GDT_DOSFREEMODULE	DOSFREEMODULE call gate
1dd8	GDT_DOSRESUMETHREAD	DOSRESUMETHREAD call gate
1de0	GDT_DOSSLEEP	DOSSLEEP call gate
1de8	GDT_DOSSUSPENDTHREAD	DOSSUSPENDTHREAD call gate
1df0	GDT_DOSLIBINIT	DOSLIBINIT call gate
1df8	GDT_REDIR	REDIR call gate
1e00	GDT_DOSCHGFILEPTR	DOSCHGFILEPTR call gate
1e08	GDT_DOSPROTECTCHGFILEPTR	DOSPROTECTCHGFILEPTR call gate
1e10	GDT_DOSCLOSE	DOSCLOSE call gate
1e18	GDT_DOSPROTECTCLOSE	DOSPROTECTCLOSE call gate
1e20	GDT_DOSDELETE	DOSDELETE call gate
1e28	GDT_DOSDEVIOCTL	DOSDEVIOCTL call gate
1e30	GDT_DOSDEVIOCTL2	DOSDEVIOCTL2 call gate

Table 164 (Page 3 of 3). GDT Assignments

Selector	**Symbol**	**Description**
1e38	GDT_DOSDUPHANDLE	DOSDUPHANDLE call gate
1e40	GDT_DOSICOPY	DOSICOPY call gate
1e48	GDT_DOSIREAD	DOSIREAD call gate
1e50	GDT_DOSIPROTECTREAD	DOSIPROTECTREAD call gate
1e58	GDT_DOSISETRELMAXFH	DOSISETRELMAXFH call gate
1e60	GDT_DOSIWRITE	DOSIWRITE call gate
1e68	GDT_DOSIPROTECTWRITE	DOSIPROTECTWRITE call gate
1e70	GDT_DOSMOVE	DOSMOVE call gate
1e78	GDT_DOSOPEN	DOSOPEN call gate
1e88	GDT_MSSTACK	
1e90	GDT_OS2LDR	os2ldr′s data
1e98	GDT_NWDTSS	NMI TSS
1ea0	GDT_NWDSTACK	NMI Stack Selector
1ea8	GDT_R0CSC	R0 Code Selector for Init DDs

4.5 Standard LDT Assignments

The following table lists the LDT assignments that are defined by the system.

Table 165. Standard LDT Assignments

Selector	**Description**
7	Read/Only access to the current LDT
dff4	Read/Only access to the current Global Information Segment
dff7	Read/Only access to the current Local Information Segment

4.6 VM System Object Owner IDs

System objects are a reserved range of **hobs** used to attribute ownership of virtual memory objects to system components. System object IDs have no corresponding VMOB.

The following table lists the system object IDs. The names shown are those displayed by the Kernel Debugger and Dump Formatter when formatting VMOB structures:

Table 166 (Page 1 of 6). System Object IDs

Name	**ID**	**Description**
lielist	0xff2d	LDR LieLists
demversion	0xff2e	DEM fake version entries
vmbmapd	0xff2f	VM Arena Bitmap Directory
npipenpn	0xff30	Named pipe NPN segment
npipenp	0xff31	Named pipe NP segment
reqpkttcb	0xff32	DD TCB request packets
reqpkt2	0xff33	DD strat2 request packets
spldevrmp	0xff34	Spool Dev RMP segment
chardevrmp	0xff35	Char Dev RMP segment
syssemrmp	0xff36	System Semaphore RMP segment
romdata	0xff37	ROM data
libpath	0xff38	LDR LibPath
jfnflags	0xff39	JFN flags
jfntable	0xff3a	JFN table
ptouvirt	0xff3b	PhysToUVirt
tkr3stack	0xff3c	Ring 3 stack
tkr2stack	0xff3d	Ring 2 stack
tkenv	0xff3e	User Environment
tktib	0xff3f	Thread Information Block
reqpkt1	0xff40	DD strat1 request packets
allocphys	0xff41	Allocated via DevHlp AllocPhys
khbdon	0xff42	Unusable donated heap page owner
krhrw1m	0xff43	Resident R/W 1Meg mem heap owner
krhro1m	0xff44	Resident R/W 1Meg mem heap owner
mmph	0xff45	dekko mapped memory
pageio	0xff46	pageio per-swap-file save block
fsreclok	0xff47	record lock record owner
		File System Drivers
fsd1	0xff48	FSD 1
fsd2	0xff49	FSD 2
fsd3	0xff4a	FSD 3
fsd4	0xff4b	FSD 4

Table 166 (Page 2 of 6). System Object IDs

Name	**ID**	**Description**
fsd5	0xff4c	FSD 5
fsd6	0xff4d	FSD 6
fsd7	0xff4e	FSD 7
fsd8	0xff4f	FSD 8 and subsequent
		Device Drivers
dd1	0xff50	device driver 1
dd2	0xff51	device driver 2
dd3	0xff52	device driver 3
dd4	0xff53	device driver 4
dd5	0xff54	device driver 5
dd6	0xff55	device driver 6
dd7	0xff56	device driver 7
dd8	0xff57	device driver 8
dd9	0xff58	device driver 9
dd10	0xff59	device driver 10
dd11	0xff5a	device driver 11
dd12	0xff5b	device driver 12
dd13	0xff5c	device driver 13
dd14	0xff5d	device driver 14
dd15	0xff5e	device driver 15
dd16	0xff5f	device driver 16 and subsequent
		Miscellaneous Owners
fsclmap	0xff60	cluster map owner
cdsrmp	0xff61	Current Directory Structure RMP seg
tom	0xff62	Timeout Manager
abios	0xff63	Advanced BIOS
cache	0xff64	cache
dbgdcb	0xff65	DBG Debug Control Block
dbgkdb	0xff66	DBG Kernel Debug Block
dbgwpcb	0xff67	DBP Watch Point Control Block
demsft	0xff68	DEM SFT array (for FCBs)
demfonto	0xff69	DEM font offsets
demfont	0xff6a	DEM font data
devhlp	0xff6b	allocated via devhlp AllocPhys
discard	0xff6c	discardable, zero fill object
doshlp	0xff6d	DosHelp segment
dyndtgp	0xff6e	DYN trace point parm block
dyndto	0xff6f	dynamic trace point
dyndtot	0xff70	tmp dynamic trace info
dynmtel	0xff71	DYN MTE dynamic trace link

Table 166 (Page 3 of 6). System Object IDs		
Name	**ID**	**Description**
emalloc	0xff72	EM86 malloc()
emtss	0xff73	EM86 TSS
device	0xff74	installed device driver
infoseg	0xff75	infoseg (local or global)
initmsg	0xff76	INIT saved message
init	0xff77	generic init-time only
intdirq	0xff78	INT IRQ info
intstack	0xff79	interrupt stack
iopllist	0xff7a	List of modules with IOPL
kdbalias	0xff7b	Kernel debugger alias
kdbsym	0xff7c	Kernel debugger symbol
kmhook	0xff7d	KM hook info
ksem	0xff7e	KSEM semaphore
lbdd	0xff7f	loadable base device driver
lid	0xff80	ABIOS logical identifier
monitor	0xff81	monitor segment
mshare	0xff82	named-shared
mshrmp	0xff83	RMP having mshare records
nmi	0xff84	non maskable interrupt
npx	0xff85	287/387 save area
orphan	0xff86	orphaned segment
prof	0xff87	profile support
ptogdt	0xff88	Allocated via dh_allocateGDTSelector
ptovirt	0xff89	PhysToVirt
puse	0xff8a	Page Usage
pusetmp	0xff8b	tmp Page Usage
perfview	0xff8c	Perfview
qscache	0xff8d	QuerySysInfo cache
ras	0xff8e	RAS segment
resource	0xff8f	Resource BMP segment
sysserv	0xff90	system service
timer	0xff91	timer services segment
traphe	0xff92	TRAP Hard Error
		File System Owners
fsbuf	0xff93	file system buffer
cdevtmp	0xff94	Char DEV TMP
fsc	0xff95	FSC segment
dpb	0xff96	DPB
eatmp	0xff97	fat EA TMP
fatsrch	0xff98	fat search segment

Table 166 (Page 4 of 6). System Object IDs

Name	**ID**	**Description**
gnotify	0xff99	FindNotify global segment
pnotify	0xff9a	FindNotify private segment
fsh	0xff9b	installable file sys helper
ifs	0xff9c	installable file system
mfsd	0xff9d	mini file system
mft	0xff9e	master file table
npipebuf	0xff9f	Named pipe I/O buffer segment
pipe	0xffa0	pipe
sft	0xffa1	system file table
vpb	0xffa2	volume parameter block
		Loader Owners
ldcache	0xffa3	Loader Instance Data Cache
ldrdld	0xffa4	LDR Dynamic Load record
invalid	0xffa5	Cache being made
ldrmte	0xffa6	mte
ldrpath	0xffa7	LDR MTE path
ldrnres	0xffa8	LDR non-resident names
prot16	0xffa9	Protect 16 list
		Boot Loader and Kernel Owners
os2krnl	0xffaa	os2krnl load image
os2ldr	0xffab	os2ldr load image
ripl	0xffac	Remote IPL (remote boot)
		Page Manager Owners
pgalias	0xffad	Temporary page manager aliases
pgbuf	0xffae	PG loader and swapper buffer
pgcrpte	0xffaf	PG Compat. region page table
dbgalias	0xffb0	debugger alias pte
pgdir	0xffb1	PG Page directory
pgkstack	0xffb2	kernel stack region
pgvp	0xffb3	VP array
pgpf	0xffb4	PF array
pgprt	0xffb5	Page Range Table
pgsyspte	0xffb6	PG System page tables
		Selector Manager Owners
gdt	0xffb7	SEL GDT
selheap	0xffb8	Selector-mapped heap block
ldt	0xffb9	SEL LDT
lock	0xffba	SEL Lock
selnop	0xffbb	NO-OP Locks
seluvirt	0xffbc	SEL UVIRT mapping

Table 166 (Page 5 of 6). System Object IDs		
Name	**ID**	**Description**
		Semaphore Owners
semmisc	0xffbd	SEM Miscellaneous
semmuxq	0xffbe	SEM Mux Queue
semopenq	0xffbf	SEM Open Queue
semrec	0xffc0	SEM SemRecord
semstr	0xffc1	SEM string
semstruc	0xffc2	SEM Main structure
semtable	0xffc3	SEM Private/Shared table
		Swapper Owners
smdfh	0xffc4	SM Disk Frame Heap
smsfn	0xffc5	SM SFN array
smsf	0xffc6	SM Swap Frame
		Tasking Owners
tkextlst	0xffc7	TK Exit List record
tkkmreg	0xffc8	TK dispatch (KM) registers
tklibif	0xffc9	TK LibInit Free Notification record
tklibi	0xffca	TK LibInit record
ptda	0xffcb	TK PTDA
tcb	0xffcc	TK TCB
tsd	0xffcd	TK TSD
		VDD, VDH, VDM Owners
vddblkh	0xffce	VDD block header
vddblk	0xffcf	VDD memory block
vddcfstr	0xffd0	VDD config.sys string
vddctmp	0xffd1	VDD creation tmp allocation
vddep	0xffd2	VDD Entry Point
vddheaph	0xffd3	VDD heap header
vddheap	0xffd4	heap objects to load VDDs
vddhook	0xffd5	VDD hook
vddla	0xffd6	VDD Linear Arena header
vddlr	0xffd7	VDD Linear arena Record
vddmod	0xffd8	VDD module record
vddopen	0xffd9	open VDD record
vddpddep	0xffda	VDD PDD Entry Point
vddproc	0xffdb	VDD procedure record
vddstr	0xffdc	VDD string
vdhfhook	0xffdd	VDH fault hook
vdhalloc	0xffde	VDH services resident memory
vdhswap	0xffdf	VDH services swappable memory
vdmalias	0xffe0	VDM Alias

Table 166 (Page 6 of 6). System Object IDs

Name	ID	Description
		Virtual Memory Manager Owners
vmah	0xffe1	VM arena header
vmal	0xffe2	VM Alias Record
vmar	0xffe3	VM Arena Record
vmbmap	0xffe4	VM Location Bitmap
vmco	0xffe5	VM Context Record
vmdead	0xffe6	VM Dead Object
vmhsh	0xffe7	VM Location Hash Table
vmkrhb	0xffe8	VM *UNKNOWN* busy KRHB
vmkrhf	0xffe9	VM free KRHB
vmkrhl	0xffea	VM end KRHB
vmkrhro	0xffeb	VM Public Kernel Resident R/O Heap
vmkrhrw	0xffec	VM Public Kernel Resident R/W Heap
vmkshd	0xffed	VM Swappable Heap Descriptor
vmkshro	0xffee	VM Public Kernel Swappable R/O Heap
vmkshrw	0xffef	VM Public Kernel Swappable R/W Heap
vmllock	0xfff0	VM long term lock manager
vmob	0xfff1	VM Object Record
vmsgs	0xfff2	VM Screen Group Switch record
vmbmp16	0xfff3	VM Temp buf (BMP16)
shrind	0xfff4	reserved for shared indicator
give	0xfff5	giveable segment
get	0xfff6	gettable segment
giveget	0xfff7	giveable and gettable segment
preload	0xfff8	Loader′s preload object

4.7 DevHlp Function Cross-Reference

The following table is a cross-reference for DevHlp function names with request code. The request code is loaded into the **DL** register before calling Device_Help.

Table 167 (Page 1 of 3). DevHlp Function Cross-Reference

Function Name	**Code**	**Description**
DevHlp_SchedClock	0x0	Called each timer tick
DevHlp_DevDone	0x1	Device I/O complete
DevHlp_DevDone	0x1	Device I/O complete
DevHlp_Yield	0x2	yield CPU if resched set
DevHlp_TCYield	0x3	yield to time critical task
DevHlp_ProcBlock	0x4	Block on event
DevHlp_ProcRun	0x5	Unblock process
DevHlp_SemRequest	0x6	claim a semaphore
DevHlp_SemClear	0x7	release a semaphore
DevHlp_SemHandle	0x8	obtain a semaphore handle
DevHlp_PushRequest	0x9	Push the request
DevHlp_PullRequest	0xA	Pull next request from Q
DevHlp_PullParticular	0xB	Pull a specific request
DevHlp_SortRequest	0xC	Push request in sorted order
DevHlp_AllocReqPacket	0xD	allocate request packet
DevHlp_FreeReqPacket	0xE	free request packet
DevHlp_QueueInit	0xF	Init/Clear char queue
DevHlp_QueueFlush	0x10	flush queue
DevHlp_QueueWrite	0x11	Put a char in the queue
DevHlp_QueueRead	0x12	Get a char from the queue
DevHlp_Lock	0x13	Lock segment
DevHlp_Unlock	0x14	Unlock segment
DevHlp_PhysToVirt	0x15	convert physical address to virtual
DevHlp_VirtToPhys	0x16	convert virtual address to physical
DevHlp_PhysToUVirt	0x17	convert physical to LDT
DevHlp_AllocPhys	0x18	allocate physical memory
DevHlp_FreePhys	0x19	free physical memory
DevHlp_SetROMVector	0x1A	set a ROM service routine vector
DevHlp_SetIRQ	0x1B	set an IRQ interrupt
DevHlp_UnSetIRQ	0x1C	unset an IRQ interrupt
DevHlp_SetTimer	0x1D	set timer request handler
DevHlp_ResetTimer	0x1E	unset timer request handler
DevHlp_MonitorCreate	0x1F	create a monitor
DevHlp_Register	0x20	install a monitor
DevHlp_DeRegister	0x21	remove a monitor

Table 167 (Page 2 of 3). DevHlp Function Cross-Reference

Function Name	**Code**	**Description**
DevHlp_MonWrite	0x22	pass data records to monitor
DevHlp_MonFlush	0x23	remove all data from stream
DevHlp_GetDOSVar	0x24	Return pointer to DOS variable
DevHlp_SendEvent	0x25	an event occurred
DevHlp_ROMCritSection	0x26	ROM Critical Section
DevHlp_VerifyAccess	0x27	Verify access to memory
DevHlp_RAS	0x28	Put info in RAS trace buffer
DevHlp_ABIOSGetParms	0x29	Get ABIOS Calling Parms
DevHlp_AttachDD	0x2A	Attach to a device driver
DevHlp_InternalError	0x2B	Signal an internal error
DevHlp_ModifyPriority	0x2C	Undocumented (used by PM)
DevHlp_AllocGDTSelector	0x2D	Allocate GDT Selectors
DevHlp_PhysToGDTSelector	0x2E	Convert phys addr to GDT sel
DevHlp_RealToProt	0x2F	Change from real to protected mode
DevHlp_ProtToReal	0x30	Change from protected to real mode
DevHlp_EOI	0x31	Send EOI to PIC
DevHlp_UnPhysToVirt	0x32	mark completion of PhysToVirt
DevHlp_TickCount	0x33	modify timer
DevHlp_GetLIDEntry	0x34	Obtain Logical ID
DevHlp_FreeLIDEntry	0x35	Release Logical ID
DevHlp_ABIOSCall	0x36	Call ABIOS
DevHlp_ABIOSCommonEntry	0x37	Invoke Common Entry Point
DevHlp_GetDeviceBlock	0x38	Get ABIOS Device Block
DevHlp_RegisterStackUsag	0x3A	Register for stack usage
DevHlp_LogEntry	0x3B	Place data in log buffer
DevHlp_VideoPause	0x3C	Video pause on/off - D607
DevHlp_Save_Message	0x3D	Save msg in SysInit Message Table
DevHlp_SegRealloc	0x3E	Realloc DD protect mode segment
DevHlp_PutWaitingQueue	0x3F	Put I/O request on waiting queue
DevHlp_GetWaitingQueue	0x40	Get I/O request from waiting queue
DevHlp_PhysToSys	0x41	Address conversion for the AOX
DevHlp_PhysToSysHook	0x42	Address conversion for the AOX
DevHlp_RegisterDeviceClass	0xEQU	43 Register DC entry point
DevHlp_RegisterPDD	0x50	Register PDD entry point with VDM manager for later PDD-VDD communication
DevHlp_RegisterBeep	0x51	register PTD beep service entry point with kernel
DevHlp_Beep	0x52	preempt beep service via PTD
DevHlp_FreeGDTSelector	0x53	Free allocated GDT selector

Table 167 (Page 3 of 3). DevHlp Function Cross-Reference

Function Name	**Code**	**Description**
DevHlp_PhysToGDTSel	0x54	Convert Phys Addr to GDT sel with given access
DevHlp_VMLock	0x55	Lock linear address range
DevHlp_VMUnlock	0x56	Unlock address range
DevHlp_VMAlloc	0x56	Allocate memory
DevHlp_VMFree	0x58	Free memory or mapping
DevHlp_VMProcessToGlobal	0x59	Create global mapping to process memory
DevHlp_VMGlobalToProcess	0x5A	Create process mapping to global memory
DevHlp_VirtToLin	0x5B	Convert virtual address to linear
DevHlp_LinToGDTSelector	0x5C	Convert linear address to virtual
DevHlp_GetDescInfo	0x5D	Return descriptor information
DevHlp_LinToPageList	0x5E	build pagelist array from lin addr
DevHlp_PageListToLin	0x5F	map page list array to lin addr
DevHlp_PageListToGDTSelector	0x60	map page list array to GDT sel.
DevHlp_RegisterTmrDD	0x61	Register TMR Device Driver.
DevHlp_RegisterPerfCtrs	0x62	Register device driver perf. ctrs (PVW).
DevHlp_AllocateCtxHook	0x63	Allocate a context hook
DevHlp_FreeCtxHook	0x64	Free a context hook
DevHlp_ArmCtxHook	0x65	Arm a context hook
DevHlp_VMSetMem	0x66	commit/decommit memory
DevHlp_OpenEventSem	0x67	open an event semaphore
DevHlp_CloseEventSem	0x68	close an event semaphore
DevHlp_PostEventSem	0x69	post an event semaphore
DevHlp_ResetEventSem	0x6A	reset an event semaphore
DevHlp_RegisterFreq	0x6B	register PTD freq service entry point with kernel
DevHlp_DynamicAPI	0x6C	add a dynamic API
DevHlp_ProcRun2	0x6D	Unblock process via procrun2
DevHlp_CreateInt13VDM	0x6E	Create Int13 VDM (Internal Only) OEMINT13
DevHlp_RegisterKrnlExit	0x6F	Used to capture Kernel Exits F78693
DevHlp_PMPostEventSem	0x70h	PM Post Event Semaphore

4.8 System Ordinal Cross-Reference

The following table is a cross-reference for System Entry points by Ordinal number.

Table 168 (Page 1 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
1	DOSICREATETHREAD
2	DOSCWAIT
3	DOSENTERCRITSEC
4	DOSIEXECPGM
5	DOSEXIT
6	DOSEXITCRITSEC
7	DOSEXITLIST
8	DOSGETINFOSEG
9	DOSGETPRTY
10	DOSKILLPROCESS
11	DOSSETPRTY
12	DOSPTRACE
13	DOSHOLDSIGNAL
14	DOSSETSIGHANDLER
15	DOSFLAGPROCESS
16	DOSMAKEPIPE
17	DOSISYSSEMCLEAR
18	DOSISEMREQUEST
19	DOSISYSSEMSET
20	DOSSEMSETWAIT
21	DOSISEMWAIT
22	DOSMUXSEMWAIT
23	DOSCLOSESEM
24	DOSCREATESEM
25	DOSOPENSEM
26	DOSRESUMETHREAD
27	DOSSUSPENDTHREAD
28	DOSSETDATETIME
29	DOSTIMERASYNC
30	DOSTIMERSTART
31	DOSTIMERSTOP
32	DOSSLEEP
33	DOSGETDATETIME
34	DOSALLOCSEG
35	DOSALLOCSHRSEG
36	DOSGETSHRSEG
37	DOSGIVESEG
38	DOSREALLOCSEG
39	DOSFREESEG
40	DOSALLOCHUGE

Table 168 (Page 2 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
41	DOSGETHUGESHIFT
42	DOSREALLOCHUGE
43	DOSCREATECSALIAS
44	DOSLOADMODULE
45	DOSGETPROCADDR
46	DOSFREEMODULE
47	DOSGETMODHANDLE
48	DOSGETMODNAME
49	DOSGETMACHINEMODE
50	DOSBEEP
51	DOSCLIACCESS
52	DOSDEVCONFIG
53	DOSDEVIOCTL
54	DOSSGSWITCH
55	DOSSGSWITCHME
56	DOSBUFRESET
57	DOSCHDIR
58	DOSCHGFILEPTR
59	DOSCLOSE
60	DOSDELETE
61	DOSDUPHANDLE
62	DOSFILELOCKS
63	DOSFINDCLOSE
64	DOSFINDFIRST
65	DOSFINDNEXT
66	DOSMKDIR
67	DOSMOVE
68	DOSNEWSIZE
69	DOSPORTACCESS
70	DOSOPEN
71	DOSQCURDIR
72	DOSQCURDISK
73	DOSQFHANDSTATE
74	DOSQFILEINFO
75	DOSQFILEMODE
76	DOSQFSINFO
77	DOSQHANDTYPE
78	DOSQVERIFY
79	DOSIREAD
80	DOSRMDIR

Table 168 (Page 3 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
81	DOSSELECTDISK
82	DOSSETFHANDSTATE
83	DOSSETFILEINFO
84	DOSSETFILEMODE
85	DOSSETMAXFH
86	DOSSETVERIFY
87	DOSIWRITE
88	DOSSYSTEMSERVICE
89	DOSSETVEC
90	DOSSYSTRACE
91	DOSGETENV
92	DOSGETVERSION
93	DOSQTRACEINFO
94	DOSGETPID
95	DOSOPEN2
96	DOSLIBINIT
97	DOSSETFSINFO
98	DOSQPATHINFO
99	DOSDEVIOCTL2
100	DOSICANONICALIZE
101	DOSSETFGND
102	DOSSWAPTASKINIT
103	DOSREADPHYS
104	DOSSETPATHINFO
105	DOSSGSWITCHPROC2
106	STRUCHECK
107	STRURESUPDATE
108	DOSISETRELMAXFH
109	DOSIDEVIOCTL
110	DOS32FORCEDELETE
111	DOS32KILLTHREAD
112	DOSQUERYRASINFO
113	DOS32DUMPPROCESS
114	DOS32SUPPRESSPOPUPS
118	DOSOPEN2COMPT
119	DOSGETSTDA
120	DOSERROR
121	DOSGETSEG
122	DOSLOCKSEG
123	DOSUNLOCKSEG

Table 168 (Page 4 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
124	DOSSGSWITCHPROC
125	DOSIRAMSEMWAKE
126	DOSSIZESEG
127	DOSMEMAVAIL
128	DOSIRAMSEMREQUEST
129	DOSPHYSICALDISK
130	DOSGETCP
131	DOSISETCP
132	DOSGLOBALSEG
133	DOSPROFILE
134	DOSSENDSIGNAL
135	DOSHUGESHIFT
136	DOSHUGEINCR
137	DOSREAD
138	DOSWRITE
139	DOSERRCLASS
140	DOSSEMREQUEST
141	DOSSEMCLEAR
142	DOSSEMWAIT
143	DOSSEMSET
144	DOSEXECPGM
145	DOSCREATETHREAD
146	DOSSUBSET
147	DOSSUBALLOC
148	DOSSUBFREE
149	DOSREADASYNC
150	DOSWRITEASYNC
151	DOSSEARCHPATH
152	DOSSCANENV
153	DOSSETCP
154	DOSQPROCSTATUS
155	DOSGETRESOURCE
156	DOSGETPPID
157	DOSCALLBACK
158	DOSICALLBACK
159	DOSRETFORWARD
160	DOSR2STACKREALLOC
161	DOSFSRAMSEMREQUEST
162	DOSFSRAMSEMCLEAR
163	DOSQAPPTYPE

Table 168 (Page 5 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
164	DOSSETPROCCP
165	DOSDYNAMICTRACE
166	DOSQSYSINFO
167	DOSIMAKENMPIPE
168	DOSICALLNMPIPE
169	DOSICONNECTNMPIPE
170	DOSIDISCONNECTNMPIPE
171	DOSIPEEKNMPIPE
172	DOSIQNMPIPEINFO
173	DOSIQNMPHANDSTATE
174	DOSISETNMPHANDSTATE
175	DOSITRANSACTNMPIPE
176	DOSIWAITNMPIPE
177	DOSISETNMPIPESEM
178	DOSIQNMPIPESEMSTATE
179	DOSIRAWREADNMPIPE
180	DOSIRAWWRITENMPIPE
181	DOSFSATTACH
182	DOSQFSATTACH
183	DOSFSCTL
184	DOSFINDFIRST2
185	DOSMKDIR2
186	DOSFILEIO
187	DOSFINDNOTIFYCLOSE
188	DOSFINDNOTIFYFIRST
189	DOSFINDNOTIFYNEXT
190	DOSSETTRACEINFO
191	DOSEDITNAME
192	DOSLOGMODE
193	DOSLOGENTRY
194	DOSGETLOGBUFFER
195	DOSLOGREGISTER
196	DOSLOGREAD
197	DOSFINDFROMNAME
198	DOSOPLOCKRELEASE
199	DOSOPLOCKWAIT
200	DOSICOPY
201	DOSCOPY
202	DOSIQAPPTYPE
203	DOSFORCEDELETE

Table 168 (Page 6 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
204	DOSENUMATTRIBUTE
205	DOSOPLOCKSHUTDOWN
206	DOSSHUTDOWN
207	DOSGETRESOURCE2
208	DOSFREERESOURCE
209	DOS32SETMAXFH
210	DOS32SETVERIFY
211	DOS32ERRCLASS
212	DOS32ERROR
213	DOSCREATEVDM
214	DOSMAXPATHLEN
215	DOSPAGESIZE
216	DOSLOCALINFO
217	DOSGLOBALINFO
218	DOS32SETFILEINFO
219	DOS32SETPATHINFO
220	DOS32SETDEFAULTDISK
221	DOS32SETFHSTATE
222	DOS32SETFSINFO
223	DOS32QUERYPATHINFO
224	DOS32QUERYHTYPE
225	DOS32QUERYVERIFY
226	DOS32DELETEDIR
227	DOS32SCANENV
228	DOS32SEARCHPATH
229	DOS32SLEEP
230	DOS32GETDATETIME
231	DOS32DEVCONFIG
232	DOS32ENTERCRITSEC
233	DOS32EXITCRITSEC
234	DOS32EXIT
235	DOS32KILLPROCESS
236	DOS32SETPRIORITY
237	DOS32RESUMETHREAD
238	DOS32SUSPENDTHREAD
239	DOS32CREATEPIPE
240	DOS32CALLNPIPE
241	DOS32CONNECTNPIPE
242	DOS32DISCONNECTNPIPE
243	DOS32CREATENPIPE

Table 168 (Page 7 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
244	DOS32PEEKNPIPE
245	DOS32QUERYNPHSTATE
246	DOS32RAWREADNPIPE
247	DOS32RAWWRITENPIPE
248	DOS32QUERYNPIPEINFO
249	DOS32QUERYNPIPESEMSTATE
250	DOS32SETNPHSTATE
251	DOS32SETNPIPESEM
252	DOS32TRANSACTNPIPE
253	DOS32WAITNPIPE
254	DOS32RESETBUFFER
255	DOS32SETCURRENTDIR
256	DOS32SETFILEPTR
257	DOS32CLOSE
258	DOS32COPY
259	DOS32DELETE
260	DOS32DUPHANDLE
261	DOS32EDITNAME
263	DOS32FINDCLOSE
264	DOS32FINDFIRST
265	DOS32FINDNEXT
266	DOSOPENVDD
267	DOSREQUESTVDD
268	DOSCLOSEVDD
269	DOS32FSATTACH
270	DOS32CREATEDIR
271	DOS32MOVE
272	DOS32SETFILESIZE
273	DOS32OPEN
274	DOS32QUERYCURRENTDIR
275	DOS32QUERYCURRENTDISK
276	DOS32QUERYFHSTATE
277	DOS32QUERYFSATTACH
278	DOS32QUERYFSINFO
279	DOS32QUERYFILEINFO
280	DOS32WAITCHILD
281	DOS32READ
282	DOS32WRITE
283	DOS32EXECPGM
284	DOS32DEVIOCTL

Table 168 (Page 8 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
285	DOS32FSCTL
286	DOS32BEEP
287	DOS32PHYSICALDISK
288	DOS32SETCP
289	DOS32SETPROCESSCP
290	DOS32STOPTIMER
291	DOS32QUERYCP
292	DOS32SETDATETIME
293	THK32ALLOCBLOCK
294	THK32FREEBLOCK
295	THK32R3DS
296	DOS32EXITLIST
297	DOS32ALLOCPROTECTEDMEM
298	DOS32ALIASMEM
299	DOS32ALLOCMEM
300	DOS32ALLOCSHAREDMEM
301	DOS32GETNAMEDSHAREDMEM
302	DOS32GETSHAREDMEM
303	DOS32GIVESHAREDMEM
304	DOS32FREEMEM
305	DOS32SETMEM
306	DOS32QUERYMEM
307	DOS32QUERYMEMSTATE
308	DOS32OPENVDD
309	DOS32REQUESTVDD
310	DOS32CLOSEVDD
311	DOS32CREATETHREAD
312	DOS32GETINFOBLOCKS
313	DOSALLOCPROTSEG
314	DOSALLOCSHRPROTSEG
315	DOSALLOCPROTHUGE
316	DOS32DYNAMICTRACE
317	DOS32DEBUG
318	DOS32LOADMODULE
319	DOS32QUERYMODULEHANDLE
320	DOS32QUERYMODULENAME
321	DOS32QUERYPROCADDR
322	DOS32FREEMODULE
323	DOS32QUERYAPPTYPE
324	DOS32CREATEEVENTSEM

Table 168 (Page 9 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
325	DOS32OPENEVENTSEM
326	DOS32CLOSEEVENTSEM
327	DOS32RESETEVENTSEM
328	DOS32POSTEVENTSEM
329	DOS32WAITEVENTSEM
330	DOS32QUERYEVENTSEM
331	DOS32CREATEMUTEXSEM
332	DOS32OPENMUTEXSEM
333	DOS32CLOSEMUTEXSEM
334	DOS32REQUESTMUTEXSEM
335	DOS32RELEASEMUTEXSEM
336	DOS32QUERYMUTEXSEM
337	DOS32CREATEMUXWAITSEM
338	DOS32OPENMUXWAITSEM
339	DOS32CLOSEMUXWAITSEM
340	DOS32WAITMUXWAITSEM
341	DOS32ADDMUXWAITSEM
342	DOS32DELETEMUXWAITSEM
343	DOS32QUERYMUXWAITSEM
344	DOS32SUBSETMEM
345	DOS32SUBALLOCMEM
346	DOS32SUBFREEMEM
347	DOS32SUBUNSETMEM
348	DOS32QUERYSYSINFO
349	DOS32WAITTHREAD
350	DOS32ASYNCTIMER
351	DOS32STARTTIMER
352	DOS32GETRESOURCE
353	DOS32FREERESOURCE
354	DOS32SETEXCEPTIONHANDLER
355	DOS32UNSETEXCEPTIONHANDLER
356	DOS32RAISEEXCEPTION
357	DOS32UNWINDEXCEPTION
358	DOS32QUERYPAGEUSAGE
359	DOSQUERYMODFROMCS
360	DOS32QUERYMODFROMEIP
361	DOSFPDATAAREA
362	DOS32TMRQUERYFREQ
363	DOS32TMRQUERYTIME
364	DOS32ALIASPERFCTRS

Table 168 (Page 10 of 22). System Ordinal Cross-Reference

Ordinal	Entry Point
365	DOS32CONFIGUREPERF
366	DOS32DECONPERF
367	DOS32REGISTERPERFCTRS
368	DOS32QUERYSYSSTATE
369	DOS32FLATCS
370	DOS32FLATDS
371	DOS32QUERYABIOSSUPPORT
372	DOS32ENUMATTRIBUTE
373	DOS32QUERYDOSPROPERTY
374	DOS32SETDOSPROPERTY
375	DOSQUERYDOSPROPERTY
376	DOSSETDOSPROPERTY
377	DOS32PROFILE
378	DOS32SETSIGNALEXCEPTIONFOC
379	DOS32SENDSIGNALEXCEPTION
380	DOS32ENTERMUSTCOMPLETE
381	DOS32EXITMUSTCOMPLETE
382	DOS32SETRELMAXFH
383	MSGPUTMESSAGE
384	MSGTRUEGETMESSAGE
385	MSGINSMESSAGE
386	MSG32INSERTMESSAGE
387	MSG32PUTMESSAGE
388	MSG32TRUEGETMESSAGE
389	MSGIQUERYMESSAGECP
390	MSG32IQUERYMESSAGECP
391	NLSCASEMAP
392	NLSGETCOLLATE
393	NLSGETCTRYINFO
394	NLSGETDBCSEV
395	NLS32QUERYCTRYINFO
396	NLS32QUERYDBCSENV
397	NLS32MAPCASE
398	NLS32QUERYCOLLATE
399	NPIPEMAKENMPIPE
400	NPIPEQNMPIPEINFO
401	NPIPECONNECTNMPIPE
402	NPIPEDISCONNECTNMPIPE
403	NPIPEQNMPHANDSTATE
404	NPIPESETNMPHANDSTATE

Table 168 (Page 11 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
405	NPIPEPEEKNMPIPE
406	NPIPEWAITNMPIPE
407	NPIPETRANSACTNMPIPE
408	NPIPECALLNMPIPE
409	NPIPERAWREADNMPIPE
410	NPIPERAWWRITENMPIPE
411	NPIPESETNMPIPESEM
412	NPIPEQNMPIPESEMSTATE
413	HPFSSTARTLAZYWRITER
414	QUEINSTANCEDATA
415	DOS32SHUTDOWN
416	DOS32ICACHEMODULE
417	DOS32REPLACEMODULE
418	DOS32ACKNOWLEDGESIGNALEXC
419	DOS32TIB
420	DOSTMRQUERYFREQ
421	DOSTMRQUERYTIME
422	DOSREGISTERPERFCTRS
423	DOSFLATTOSEL
424	DOSSELTOFLAT
425	DOS32FLATTOSEL
426	DOS32SELTOFLAT
427	DOSIODELAYCNT
428	DOS32SETFILELOCKS
429	DOS32CANCELLOCKREQUEST
430	LOGOPEN
431	LOGCLOSE
432	LOGADDENTRIES
433	LOGGETENTRIES
434	LOGSETSTATE
435	LOGSETNAME
436	LOGQUERYSTATE
437	DOSOPENCHANGENOTIFY
438	DOSRESETCHANGENOTIFY
439	DOSCLOSECHANGENOTIFY
440	DOS32OPENCHANGENOTIFY
441	DOS32RESETCHANGENOTIFY
442	DOS32CLOSECHANGENOTIFY
443	DOSQUERYABIOSSUPPORT
444	DOS32FORCESYSTEMDUMP

Table 168 (Page 12 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
454	DOS32ALLOCTHREADLOCALMEMORY
455	DOS32FREETHREADLOCALMEMORY
460	DOS32VERIFYPIDTID
464	PTDA_LANMAN_SEC
465	PTDA_PID
466	SAS_SEL
467	TCB_OPCOOKIE
468	TCB_OPFLAGS
469	TCB_NEWFLAGS
470	TCB_USER_ID
471	TCB_PROC_ID
472	TCB_FSHARING
473	TCB_SRVATTRIB
474	TCB_ALLOWED
475	TCB_PRTCB
476	TCB_NUMBER
477	TCB_THISSFT
478	TCB_THISCDS
479	TKOPTDA
480	PTDA_CRITSEC
481	PTDA_HOLDSIGCNT
482	PTDA_PPTDAPARENT
483	PTDA_PGDATA
484	PTDA_HANDLE
485	PTDA_MODULE
486	PTDA_LDTHANDLE
487	PTDA_CODEPAGE_TAG
488	PTDA_JFN_LENGTH
489	PTDA_JFN_PTABLE
490	PTDA_JFN_FLG_PTR
491	PTDA_EXTERR_LOCUS
492	PTDA_EXTERR
493	PTDA_EXTERR_ACTION
494	PTDA_EXTERR_CLASS
495	PTDA_PPID
496	PTDA_PROCTYPE
497	PTDA_CURRTCB
498	PTDA_CURRTSD
499	PTDA_SIGNATURE
545	DOS32EXCEPTIONCALLBACK

Table 168 (Page 13 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
548	DOS32R3EXCEPTIONDISPATCHER
549	DOSLIBIDISP
550	DOSLIBIDISP16
551	DOSLIBIDISP32
552	DOSR3EXITADDR
553	DOS32R3EXITADDR
554	DOS32IREAD
556	DOS32IWRITE
565	DOSISETFILEINFO
566	DOSISETPATHINFO
569	DOSIFINDNEXT
572	DOS32QUERYRESOURCESIZE
573	DOSQUERYRESOURCESIZE
574	DOSPMSEMWAIT
575	DOSPMMUXSEMWAIT
576	THK16_UNITHUNK
577	HT16_STARTUP
580	DOS32INITIALIZEPORTHOLE
582	DOS32QUERYHEADERINFO
583	DOSINITIALIZEPORTHOLE
584	DOSQUERYHEADERINFO
585	MON32MONREAD
586	DOS32QUERYPROCTYPE
587	DOSQUERYPROCTYPE
588	MON32MONWRITE
589	DOSISIGDISPATCH
592	DOS32DLLTERMDISP
594	DOS32IRAISEEXCEPTION
597	DOS32IQUERYFHSTATE
598	DOS32ISETFHSTATE
599	DOSLDTSEL
600	DOS32R3FRESTOR
601	DOSIFINDFIRST
615	DOS32IPROTECTWRITE
617	DOSIPROTECTSETFILEINFO
618	DOS32IPROTECTSETFILEINFO
619	DOS32IPROTECTSETFHSTATE
620	DOS32IPROTECTQUERYFHSTATE
621	DOS32PROTECTSETFILEPTR
622	DOSPROTECTCLOSE

Table 168 (Page 14 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
623	DOSPROTECTFILEIO
624	DOSPROTECTFILELOCKS
625	DOSIPROTECTREAD
626	DOSIPROTECTWRITE
627	DOSPROTECTNEWSIZE
628	DOSPROTECTOPEN
629	DOSPROTECTQFHANDSTATE
630	DOSPROTECTSETFHANDSTATE
631	DOSPROTECTQFILEINFO
632	DOSPROTECTSETFILEINFO
634	DOSPROTECTCHGFILEPTR
635	DOSPROTECTENUMATTRIBUTE
636	DOS32PROTECTENUMATTRIBUTE
637	DOS32PROTECTOPEN
638	DOS32PROTECTCLOSE
639	DOS32PROTECTSETFILELOCKS
640	DOS32PROTECTSETFILESIZE
641	DOS32PROTECTREAD
642	DOS32PROTECTWRITE
643	DOS32PROTECTSETFILEINFO
644	DOS32PROTECTSETFHSTATE
645	DOS32PROTECTQUERYFHSTATE
646	DOS32PROTECTQUERYFILEINFO
647	DOS32IPROTECTREAD
649	MSGCLOSEMESSAGEFILE
650	DOSLDRDIRTYWORKER
651	DOS16LDRDIRTYWORKER
661	QUEDOS32READQUEUE
662	QUEDOS32PURGEQUEUE
663	QUEDOS32CLOSEQUEUE
664	QUEDOS32QUERYQUEUE
665	QUEDOS32PEEKQUEUE
666	QUEDOS32WRITEQUEUE
667	QUEDOS32OPENQUEUE
668	QUEDOS32CREATEQUEUE
669	SMGDOS32STARTSESSION
670	SMGDOS32SELECTSESSION
671	SMGDOS32SETSESSION
672	SMGDOS32STOPSESSION
673	SMGREGISTERNOTIFICATION

Table 168 (Page 15 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
674	QUEDOSREADQUEUE
675	QUEDOSPURGEQUEUE
676	QUEDOSCLOSEQUEUE
677	QUEDOSQUERYQUEUE
678	QUEDOSPEEKQUEUE
679	QUEDOSWRITEQUEUE
680	QUEDOSOPENQUEUE
681	QUEDOSCREATEQUEUE
682	CHRDOSSMGETME
683	CHRDOSSMFREEMEM
684	CHRDOSSMGETSGCB
685	CHRDOSSMINITSGCB
686	SMGDOSSMSGDOPOPUP
687	SMGDOSSMSWITCH
688	SMGDOSSMSERVEAPPREQ
689	SMGDOSGETTIMES
690	SMGDOSSMSETTITLE
691	SMGDOSSCRUNLOCK
692	SMGDOSSMDOAPPREQ
693	SMGDOSSTOPSESSION
694	SMGDOSSELECTSESSION
695	SMGDOSSCRLOCK
696	SMGDOSSAVREDRAWWAIT
697	SMGDOSSAVREDRAWUNDO
698	SMGDOSSMSGENDPOPUP
699	SMGDOSSETSESSION
700	SMGDOSSETMNLOCKTIME
701	SMGDOSMODEUNDO
702	SMGDOSSTARTSESSION
703	SMGDOSSMGETSTATUS
704	SMGDOSSMMODEWAIT
705	SMGDOSSMTERMINATE
706	SMGDOSSMGETAPPREQ
707	SMGDOSSMINITIALIZE
708	SMGDOSSMSTART
709	SMGDOSSMPARENTSWITCH
710	SMGDOSSMPAUSE
711	SMGDOSSMHDEINIT
712	SMGDOSSMPMPRESENT
713	SMGDOSSMREGISTERDD

Table 168 (Page 16 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
714	SMGDOSSMNOTIFYDD
715	SMGDOSSMNOTIFYDD2
716	SMGDOSSMOPENDD
717	SMGDOSSMSETSESSIONTYPE
718	CHRBASEINIT
719	MOUDOSGETPTRSHAPE
720	MOUDOSSETPTRSHAPE
721	MOUDOSGETNUMMICKEYS
722	MOUDOSGETTHRESHOLD
723	MOUDOSSHELLINIT
724	MOUDOSGETSCALEFACT
725	MOUDOSFLUSHQUE
726	MOUDOSGETNUMBUTTONS
727	MOUDOSCLOSE
728	MOUDOSSETTHRESHOLD
729	MOUDOSSETSCALEFACT
730	MOUDOSGETNUMQUEEL
731	MOUDOSDEREGISTER
732	MOUDOSGETEVENTMASK
733	MOUDOSSETEVENTMASK
734	MOUDOSOPEN
735	MOUDOSREMOVEPTR
736	MOUDOSGETPTRPOS
737	MOUDOSREADEVENTQUE
738	MOUDOSSETPTRPOS
739	MOUDOSGETDEVSTATUS
740	MOUDOSSYNCH
741	MOUDOSREGISTER
742	MOUDOSSETDEVSTATUS
743	MOUDOSDRAWPTR
744	MOUDOSINITREAL
745	KBDDOSSETCUSTXT
746	KBDDOSPROCESSINIT
747	KBDDOSGETCP
748	KBDDOSCHARIN
749	KBDDOSSETCP
750	KBDDOSLOADINSTANCE
751	KBDDOSSYNCH
752	KBDDOSREGISTER
753	KBDDOSSTRINGIN

Table 168 (Page 17 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
754	KBDDOSGETSTATUS
755	KBDDOSSETSTATUS
756	KBDDOSGETFOCUS
757	KBDDOSFLUSHBUFFER
758	KBDDOSXLATE
759	KBDDOSSWITCHFGND
760	KBDDOSSHELLINIT
761	KBDDOSCLOSE
762	KBDDOSFREEFOCUS
763	KBDDOSFREE
764	KBDDOSDEREGISTER
765	KBDDOSSETFGND
766	KBDDOSPEEK
767	KBDDOSOPEN
768	KBDDOSGETHWID
769	KBDDOSSETHWID
770	VIODOSENDPOPUP
771	VIODOSGETPHYSBUF
772	VIODOSGETANSI
773	VIODOSFREE
774	VIODOSSETANSI
775	VIODOSDEREGISTER
776	VIODOSSCROLLUP
777	VIODOSPRTSC
778	VIODOSGETCURPOS
779	VIODOSWRTCELLSTR
780	VIODOSPOPUP
781	VIODOSSCROLLRT
782	VIODOSWRTCHARSTR
783	VIODOSAVS_PRTSC
784	VIODOSSETCURPOS
785	VIODOSSRFUNBLOCK
786	VIODOSSRFBLOCK
787	VIODOSSCRUNLOCK
788	VIODOSWRTTTY
789	VIODOSSAVE
790	VIODOSGETMODE
791	VIODOSSETMODE
792	VIODOSSCRLOCK
793	VIODOSREADCELLSTR

Table 168 (Page 18 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
794	VIODOSSAVREDRAWWAIT
795	VIODOSWRTNATTR
796	VIODOSGETCURTYPE
797	VIODOSSAVREDRAWUNDO
798	VIODOSGETFONT
799	VIODOSREADCHARSTR
800	VIODOSGETBUF
801	VIODOSSETCURTYPE
802	VIODOSSETFONT
803	VIODOSHETINIT
804	VIODOSMODEUNDO
805	VIODOSSSWSWITCH
806	VIODOSMODEWAIT
807	VIODOSAVS_PRTSCTOGGLE
808	VIODOSGETCP
809	VIODOSRESTORE
810	VIODOSSETCP
811	VIODOSSHOWBUF
812	VIODOSSCROLLLF
813	VIODOSREGISTER
814	VIODOSGETCONFIG
815	VIODOSSCROLLDN
816	VIODOSWRTCHARSTRATT
817	VIODOSGETSTATE
818	VIODOSPRTSCTOGGLE
819	VIODOSSETSTATE
820	VIODOSWRTNCELL
821	VIODOSWRTNCHAR
822	VIODOSSHELLINIT
823	VIODOSASSOCIATE
824	VIODOSCREATEPS
825	VIODOSDELETESETID
826	VIODOSGETDEVICECELLSIZE
827	VIODOSGETORG
828	VIODOSCREATELOGFONT
829	VIODOSDESTROYPS
830	VIODOSQUERYSETIDS
831	VIODOSSETORG
832	VIODOSQUERYFONTS
833	VIODOSSETDEVICECELLSIZE

Table 168 (Page 19 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
834	VIODOSSHOWPS
835	VIODOSGETPSADDRESS
836	VIODOSQUERYCONSOLE
837	VIODOSREDRAWSIZE
838	VIODOSGLOBALREG
839	XVIODOSSETCASTATE
840	XVIODOSCHECKCHARTYPE
841	XVIODOSDESTROYCA
842	XVIODOSCREATECA
843	VIOCDOSHECKCHARTYPE
844	XVIODOSGETCASTATE
845	BVSDOSMAIN
846	BVSDOSREDRAWSIZE
847	BVSDOSGETPTRDRAWNAME
848	ANSIDOSINJECT
849	ANSIDOSKEYDEF
850	ANSIDOSINTERP
851	BKSDOSMAIN
852	BMSDOSMAIN
853	MOUDOSGETHOTKEY
854	MOUDOSSETHOTKEY
855	SMGDOSSMSYSINIT
856	SMGQHKEYBDHANDLE
857	SMGQHMOUSEHANDLE
858	CHRQueueRamSem
859	CHRArray
860	CHRPIDArray
861	CHRInitialized
862	CHRArraySize
863	CHRBVSGLOBAL
864	CHRSMGINSTANCE
865	CHRBVHINSTANCE
115	THK32ALLOCMEM
116	THK32FREEMEM
117	THK32ALLOCSTACK
262	THK32FREESTACK
546	THK32STRLEN
547	THK32_UNITHUNK
578	HT16_CLEANUP
579	HT32_STARTUP

Table 168 (Page 20 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
581	HT32_CLEANUP
590	DOS32PMPOSTEVENTSEM
591	DOS32PMWAITEVENTSEM
593	DOS32PMREQUESTMUTEXSEM
595	DOS32PMWAITMUXWAITSEM
596	DOS32PM16SEMCHK
866	THK32ALIASMEM
867	THK32FREEALIAS
868	THK32ALLOCVARLEN
869	THK32HANDLEBOUNDARY
870	THK32HANDLESTRING
871	THK32DEALLOC
872	THK32XHNDLR
873	DOS32SETEXTLIBPATH
874	DOS32QUERYEXTLIBPATH
875	DOS32PM16SEMRST
876	DOS32SYSCTL
998	DOSSETEXTLIBPATH
999	DOSQUERYEXTLIBPATH
1000	T32EXITLIST
1001	T32ALLOCPROTECTEDMEM
1002	T32ALIASMEM
1003	T32ALLOCMEM
1004	T32ALLOCSHAREDMEM
1005	T32GETNAMEDSHAREDMEM
1006	T32GETSHAREDMEM
1007	T32GIVESHAREDMEM
1008	T32FREEMEM
1009	T32SETMEM
1010	T32QUERYMEM
1011	T32QUERYMEMSTATE
1012	T32OPENVDD
1013	T32REQUESTVDD
1014	T32CLOSEVDD
1015	T32CREATETHREAD
1016	T32DYNAMICTRACE
1017	T32DEBUG
1018	T32QUERYPROCADDR
1019	T32CREATEEVENTSEM
1020	T32OPENEVENTSEM

Table 168 (Page 21 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
1021	T32CLOSEEVENTSEM
1022	T32RESETEVENTSEM
1023	T32POSTEVENTSEM
1024	T32WAITEVENTSEM
1025	T32QUERYEVENTSEM
1026	T32CREATEMUTEXSEM
1027	T32OPENMUTEXSEM
1028	T32CLOSEMUTEXSEM
1029	T32REQUESTMUTEXSEM
1030	T32RELEASEMUTEXSEM
1031	T32QUERYMUTEXSEM
1032	T32CREATEMUXWAITSEM
1033	T32OPENMUXWAITSEM
1034	T32CLOSEMUXWAITSEM
1035	T32WAITMUXWAITSEM
1036	T32ADDMUXWAITSEM
1037	T32DELETEMUXWAITSEM
1038	T32QUERYMUXWAITSEM
1039	T32QUERYSYSINFO
1040	T32WAITTHREAD
1041	T32GETRESOURCE
1042	T32FREERESOURCE
1043	T32EXCEPTIONCALLBACK
1044	T32QUERYPAGEUSAGE
1045	T32FORCESYSTEMDUMP
1046	TI32ASYNCTIMER
1047	TI32STARTTIMER
1048	T32QUERYABIOSSUPPORT
1049	T32QUERYMODFROMEIP
1050	T32ALIASPERFCTRS
1051	T32CONFIGUREPERF
1052	T32DECONPERF
1053	T32REGISTERPERFCTRS
1054	T32QUERYSYSSTATE
1055	T32IREAD
1056	T32IWRITE
1057	T32TMRQUERYFREQ
1058	T32TMRQUERYTIME
1059	T32IMONREAD
1060	T32IMONWRITE

Table 168 (Page 22 of 22). System Ordinal Cross-Reference

Ordinal	**Entry Point**
1061	T32QUERYRESOURCESIZE
1062	T32PROFILE
1063	T32SETSIGNALEXCEPTIONFOC
1064	T32SENDSIGNALEXCEPTION
1065	T32STARTTIMER
1066	T32STOPTIMER
1067	T32ASYNCTIMER
1068	T32INITIALIZEPORTHOLE
1069	T32QUERYHEADERINFO
1070	T32QUERYPROCTYPE
1071	T32IEXITMUSTCOMPLETE
1072	T32ICACHEMODULE
1073	T32DLLTERM
1074	T32IRAISEEXCEPTION
1075	T32ACKNOWLEDGESIGNALEXC
1076	T32QUERYDOSPROPERTY
1077	T32SETDOSPROPERTY
1078	T32SETFILELOCKS
1079	T32CANCELLOCKREQUEST
1080	T32KILLTHREAD
1081	TQUERYRASINFO
1082	T32DUMPPROCESS
1083	T32SUPPRESSPOPUPS
1084	T32IPROTECTWRITE
1085	T32PROTECTSETFILELOCKS
1086	T32IPROTECTREAD
1087	T32PMPOSTEVENTSEM
1088	T32PMWAITEVENTSEM
1089	T32PMREQUESTMUTEXSEM
1090	T32PMWAITMUXWAITSEM
1091	T32PM16SEMCHK
1092	T32ALLOCTHREADLOCALMEMORY
1093	T32FREETHREADLOCALMEMORY
1094	T32SETEXTLIBPATH
1095	T32QUERYEXTLIBPATH
1096	T32PM16SEMRST
1097	T32VERIFYPIDTID
1098	T32SYSCTL

4.9 OS/2 FixPak to Build Level Cross-Reference

The following table cross-references some of the public FixPaks and General Availability versions of OS/2 with their internal kernel build level.

Note: Some FixPaks use the same kernel build level when updates are confined to modules other than OS2KRNL.

Table 169. FixPak to Build Level Cross-Reference

Version	**Build**
2.11 GA	6.617
Warp GA	8.162
Warp Connect	8.209
Warp for Windows Connect	8.200
XR_W005	8.213B
XR_W007	8.230
XR_W008	8.230
XR_W009	8.234
XR_W010	8.234
XR_W011	8.235
XR_W012	8.236
XR_W013	8.237
XR_W014	8.238
XR_W016	8.240
XR_W017	8.240
XR_A076	6.653
XR_A080	6.653
XR_A090	6.656
XR_A092	6.658
XR_A095	6.661
XR_A096	6.660
XR_B097	6.664
XR_B098	6.665
XR_B099	6.667
XR_B100	6.668
XR_B101	6.669
XR_B102	6.670
XR_B103	6.671
XR_B104	6.672
XR_B105	6.673

Index

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
OS/2 Debugging Handbook - Volume IV
System Diagnostic Reference
February 1996

Publication No. SG24-4643-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

____

____

What other topics would you like to see in this Bulletin?

____

____

What other Technical Bulletins would you like to see published?

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

______________________________ ______________________________
Name Address

______________________________ ______________________________
Company or Organization

______________________________ ______________________________
Phone No.

ITSO Technical Bulletin Evaluation **RED000**
SG24-4643-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 626C, Building 014-1
Internal Zip 5220
1000 Northwest 51st Street
Boca Raton, Florida
USA 33431-1328

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SG24-4643-00