UNISYS

BTOS Protected Mode Operating System Server (PMOSS)

Installation Guide

Relative to Release Level 1.0

February 1987

Priced Item

5026172

UNISYS

BTOS Protected Mode Operating System Server (PMOSS)

Installation Guide

Relative to Release Level 1.0 — February 1987

Priced Item — 5026172

Unisys believes that the software described in this manual is accurate, and much care has been taken in its preparation.

The customer's attention is drawn to the provisions of the Trade Practices Act 1974 (as amended) ('the Act') which imply conditions and warranties into certain contracts for the supply of goods and services. Where such conditions and warranties are implied Unisys liability shall be limited (subject to the provision of Section 68A of the Act) to the replacement or repair of the goods or the supply of equivalent goods.

The customer should exercise care to assure that use of this manual and the software will be in full compliance with the laws, rules and regulations of the jurisdiction in which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the CLASS specified as 2 (S.SW: System Software), the type specified as 1 (F.T.R.), and the product specified as the 7-digit form number of the manual (for example, 5023906).

About This Guide

The Protected Mode Operating System Server (PMOSS) allows you to run programs in protected mode utilizing memory above one megabyte on the B 28 and B 38 workstations. This document contains introductory, installation, and error code information about PMOSS.

Who Should Use This Guide

This guide will help you if you are an experienced programmer or System Adminstrator supporting programmers who write programs that run in protected mode. The procedures are easier to perform if you are familiar with BTOS and your system's configuration.

How This Guide is Arranged

This guide has four main parts.

- Section 1 provides overview information about the product.
- Section 2 provides installation information and procedures for installing PMOSS on your system.
- Section 3 contains additional information for writing programs that run in protected mode.
- Section 4 provides information about operating system restrictions when using PMOSS.
- Appendix A contains errors codes and explanations of their meanings.

Configuration

Section 2 contains the configuration and memory requirements you must know before you install PMOSS. It also contains procedures for the actual installation.

Reference Material

This guide contains an appendix with error code information, a glossary, and an index.

Related Product Information

For information about the operating system, refer to the *BTOS Reference Manual.*

For information about writing programs that will run in protected mode, refer to the *BTOS Protected Mode Programming Guide.*

For an explanation of BTOS Executive Commands, refer to the *BTOS Standard Software Operations Guide.*

Contents

Tables

Overview

This guide describes the Protected Mode Operating System Server, PMOSS.

PMOSS is an extension to the BTOS operating system which adds the capability of running programs in protected mode. It can be installed on B 28 and B 38 systems running multi-partition BTOS 8.0.

PMOSS loads programs capable of B 28 or B 38 protected mode execution and allows them to run in this mode transparently to the resident portion of the operating system, which is running in real mode.

Why Protected Mode?

One of the principal advantages of protected mode is that it allows the use of memory above 1 megabyte. Protected mode programs are loaded with their code segments at addresses above 1 megabyte (protected mode memory) and their data segments below 1 megabyte (real mode memory). This savings is particularly noteworthy on workstations where many servers, or large servers, are installed. The use of PMOSS allows certain configurations of servers and application programs that would not be possible with only one megabyte of memory.

Another important benefit is that programs running in protected mode are unlikely to damage the operating system or other programs if they malfunction, because their code and data segments are encapsulated within a private address space. Protected mode provides a development environment in which software errors are detected earlier, the code in error is more rapidly identified, and there is greater confidence that a program which appears to run properly is, in fact, correct. These advantages shorten development time and increase software reliability. PMOSS consists of some of the components of a protected mode operating system, packaged in a way that allows it to be installed as a server for use with a real-mode operating system.

Architecture

PMOSS uses proprietary hardware/software for switching between protected and real modes of operation. Switching modes involves approximately the same overhead as switching between processes in real mode, and occurs in response to the same kinds of events (for example, interrupts, or requests for system services).

The interrupt system remains fully operational while the processor is executing in either mode. Interrupts are routed to the appropriate service routine by switching modes, if necessary.

Only programs which abide by certain rules governing protected mode software can execute in protected mode. These rules are described in the *BTOS Protected Mode Programming Guide*.

Provided the program conforms to these rules, it can be executed in either real or protected mode. The same run file will execute in either mode if it is linked with the BIND command to produce a version 6 run file format. (Refer to the BTOS Linker/Librarian Programming Reference Manual for more information.) The run file runs in real mode unless loaded on a B 28 or B 38 system with PMOSS installed and enabled.

Installation of PMOSS

You must install PMOSS before installing any protected mode servers or clients of the SPA Mover interface. The following is the recommended installation sequence:

- Install PMOSS.
- Install other servers, either protected or real mode servers, in any order.
- Install the RamDisk Server, if you are using SPA (refer to the BTOS System Performance Accelerator (SPA) Installation Guide for more information.)
- Install Context Manager, if desired (for more information, refer to the BTOS Context Manager Administration Guide.)

The hard disk memory requirement for PMOSS files is approximately 229 sectors.

Installing PMOSS

You must have a B 28 or B 38 to install PMOSS.

To install PMOSS, use the following procedure:

1 At the Executive level, enter the command, INSTALL PROTECTED MODE, and press **Return**.

The following form appears:

```
Command Install Protected Mode

    [Maximum number of protected processes]
    [Maximum number of concurrent protected mode messages]
    [Maximum number of Mover Server segments]
```

The default values for the installation parameters are 10 protected processes, 100 and 25 small and large messages, and 200 mover server segments. It is not necessary to enter values unless you have a larger configuration of protected mode programs that exceeds the default resource allocations. This normally causes one of the error codes listed under Error Codes to be issued.

2 Press **GO** to execute the command.

The INSTALL PROTECTED MODE command executes the run file [Sys]<Sys>PMAgent.run, which can also be executed in a batch job stream (such as the [Sys]<Sys>SysInit.jcl file).

Note: PMOSS does not install and exits without error if you try to install it on a system that does not have a B 28 or B 38 microprocessor, more than 1 megabyte of memory, and the multipartition version of BTOS 8.0.

Use of PMOSS with SPA

PMOSS is compatible with the System Performance Accelerator (SPA), a facility for caching disk files in protected mode memory on a B 28 or B 38 workstation. (Refer to BTOS System Performance Acceleration (SPA) Installation Guide for more information). PMOSS and SPA version 2.0 (or later) may be used concurrently, if the following conditions are met:

- You must install PMOSS (PmAgent.run) before the SPA RamDisk Server (RamDisk.run).
- Do not install the SPA Memory Mover Server (Mover.run) when using PMOSS. PMOSS replaces the functionality of this server, and installing both will cause a conflict. PMOSS supports the same request interface provided by Mover.run, so that clients of the Mover Server (including RamDisk.run and the Context Manager) will run without Mover.run when PMOSS is installed. Use Mover.run only when you are not using PMOSS.

Loading Protected Mode Programs

Run files that you link with the BIND command are automatically flagged executable-in-protected-mode if PMOSS is installed and enough memory is available. This occurs because PMOSS filters the internal BTOS loader requests (LoadTask) and intervenes when it detects a protected mode run file. There is, therefore, no need for a special command to execute protected mode programs.

Disabling and Re-Enabling Protected Mode

You use the DISABLE PROTECTED MODE command to prevent subsequently executed programs from running in protected mode. Once this command is performed, subsequently loaded programs will run in real mode even if they are capable of protected mode operation. However, previously loaded protected mode programs that still may be executing (such as servers) will continue to do so in protected mode. The effect of this command is to prevent PMOSS from starting any more protected programs.

To disable protected mode, use the following procedure:

1 At the Executive level, enter DISABLE PROTECTED MODE.

2 Press GO.

All programs loaded after this point will run in real mode.

To re-enable protected mode, use the following procedure:

1 At the Executive level, enter ENABLE PROTECTED MODE.

2 Press GO.

PMOSS will now load protected mode programs again.

The run files [Sys]<Sys>DisablePmAgent.run and [Sys]<Sys>EnablePmAgent.run implement these commands and can be run from batch files (such as [Sys]<Sys>SysInit.jcl).

Installing PMOSS on Your Workstation

You must install PMOSS only on systems with a hard disk or workstations that are clustered to a master with a hard disk.

To install PMOSS on your workstation, use the following procedure:

1 Place the PMOSS installation diskette in drive [f0].

2 Enter the SOFTWARE INSTALL command at the Executive level.

3 Press GO.

Installing PMOSS on Your XE 520 System

To install PMOSS on your XE 520, use the following procedure:

1 Sign on to the system from the master workstation.
2 Place the PMOSS installation diskette in drive [f0].
3 Enter the SOFTWARE INSTALL command at the Executive level.
4 Press GO.

Workstation Environment

On a B 28 or B 38 workstation, PMOSS requires a multipartition configuration of the operating system.

The Debugger can be used with both real mode and protected mode programs. Use of the Debugger with protected mode programs is described in the *BTOS Protected Mode Programming Guide.*

Compatibility with SPA Mover Server Interface

PMOSS is compatible with the use of memory above 1 megabyte by the RamDisk Server, Context Manager, and other clients of the SPA Mover Server request interface.

PMOSS is compatible with the SPA RamDisk Cacher version 2.0 and later.

PMOSS requires Context Manager version 3.0, (or later) when the Context Manager's swap–file–in–RAM feature is used.

PMOSS replaces the SPA Mover.run server. PMOSS implements the identical set of request interfaces provided by Mover.run as described in the *BTOS Protected Mode Programming Guide.*

You may write client programs that use the Mover Server interface in a manner that will work with the SPA Mover.run Server, but they will not run with a PMOSS implementation of the same interface. Consult the *BTOS Protected Mode Programming Guide* for details of the PMOSS implementation.

XE 520 Environment

PMOSS is not executable on the XE 520 itself. However, it is possible to install PMOSS on the hard disk of an XE 520 for use by diskless B 28 or B 38 cluster workstations. The appropriate workstation operating system and [Sys]<Sys>DebuggerN.sys must also be present in the XE 520 [Sys]<Sys> directory where the workstation will use them when it boots.

Hardware Configurations

PMOSS can be installed on B 28 or B 38 systems that have more than 1 megabyte of memory. Suitable hardware includes B 28 or B 38 standalone, master, and cluster workstations with or without local hard disks.

Special Hardware Requirements

Check to see that memory expansion cartridges are firmly seated in the B 28 or B 38 workstation. PMOSS and the SPA are software components that test or use this expansion memory; in particular, the boot ROM tests only the first megabyte of RAM. PMOSS tests the remaining memory during installation. Memory errors are most often due to loose, improperly installed cartridges.

Verify that the amount of memory displayed by PMOSS at installation time agrees with the actual number of cartridges installed. Occasionally, a loose cartridge will not cause a detectable error, but will result in missing memory.

Memory Requirements

The following text describes the available memory necessary to install PMOSS on your system.

The Server

PMAgent.run occupies a 23 KB real mode partition when you use the default command form parameters. You can reduce the size of this partition by using smaller parameters; using larger values will increase its size.

PmAgent reserves 512 bytes of real mode memory for each protected mode process, multiplied by the first parameter.

The second parameter line takes two numeric values; the maximum number of small and large messages, respectively. PMOSS requires a message buffer for each request block (or other inter-process message) involving a protected mode process. Messages to or from protected mode processes require these buffers; messages between real mode processes do not.

Since message buffers are re-used, the maximums refer to the number of messages which may be simultaneously in existence. A message buffer is required for a request block from the time it is issued until its response is received. In addition, objects such as timers (TRBs or TPIBs) normally require one message buffer each while they are active. Small message buffers use 48 bytes of real mode memory each, and accommodate request blocks up to 40 bytes in size, or timers. Large message buffers cost 136 bytes each, and accomodate request blocks up to 128 bytes in size. 128 bytes is larger than the cluster permits, but is allowed for requests within a workstation. If the small parameter is unspecified the default is 100 small messages. If the large parameter is unspecified the default is one-quarter of the small parameter. If the default values are used, a total of 125 message buffers is allocated; 100 small and 25 large.

In addition to the real mode partition, PMOSS includes approximately 85 KB of protected mode code and static data (from the file [Sys]<Sys>Pmos.img). PmAgent.run loads this code and data beginning at 992 KB, thus using the 32 KB of RAM in the address range 992 KB to 1 megabyte (MB). In real mode, this 32 KB is eclipsed by the boot ROM and video RAM and is inaccessible. In protected mode, it is contiguous with the upper megabytes, and fully usable.

PMOSS uses upper memory for its dynamically created system data structures. The amount of memory consumed by these structures varies with the number and size of protected mode programs that are running.

The Maximum number of Mover Server segments parameter limits the number of Mover Server segments (distinct allocation units) that can exist simultaneously. The default is 200.

PMOSS reserves 24 bytes of protected mode memory (in the upper megabytes) times the value of this parameter for Mover Server control structures. If the SPA Mover Server interface is not used, this parameter should be set to zero. No real mode memory is reserved. The maximum possible value of this parameter is 2048.

The SPA RamDisk Cacher uses one mover segment per cached file. The Context Manager uses only a single mover segment for its swap space.

Mover Server segments are managed independently from the segments used for protected mode programs and PMOSS system data structures, so this parameter has no effect on the number of protected mode programs that you can run.

Protected Mode Programs

Programs that are to run under PMOSS must:

- conform to the rules for protected mode software (described in the *BTOS Protected Mode Programming Guide)*
- be linked by use of the BIND command using Linker version 8.0, which flags your program as executable-in-protected-mode after linking
- be linked with CTOS.Lib

Use of PMOSS

The main purpose of PMOSS is to run servers in protected mode to achieve a real mode memory savings. In this way PMOSS makes it possible for B 28 or B 38 workstations to run more than 1 megabyte of software.

A second purpose is to permit the porting of software to protected mode, and to acquire experience with protected mode programming and debugging.

Memory Requirements

Protected mode programs still require some real mode memory (in the first megabyte), for the following reasons:

- Only the code, not the data, of the program is stored in the upper megabytes, because BTOS (and other real mode servers) require direct access to the program's data in order to serve requests from it.
- Each protected mode program still requires a real mode partition and the partition data structures associated with it. Under BTOS 8.0, this overhead is 4 to 6 KB for an application, and less than 1 KB for a server once it has issued ConvertToSys.

To calculate the expected real mode memory requirement for a protected mode program, add the total size of the code segments as shown in the Linker map. You can expect the figure in the used column of the Partition Status display to be smaller by this amount when the program is run in protected mode.

SPA RamDisk Server

This server consumes protected mode memory for the files you tell it to cache. You should first install PMOSS, then any protected mode servers, and then the RamDisk Server.

Some experimentation will allow you to trim the RamDisk file list to a size that will fit in the remaining memory. The approximate amount of memory that will be consumed can be anticipated by checking the number of 512 byte sectors required by the file on disk.

Context Manager

If you plan to use the Context Manager's swap-in-file-RAM facility, trim the RamDisk Server file list still further by the number of sectors that will be required for the swap file. Refer to the BTOS Context Manager Administration Guide for more details.

Plan to leave those sectors of upper memory available because you must install the Context Manager after installing the RamDisk Server.

Protected Mode Applications (Nonserver Programs)

The memory configuration technique described above works well when only servers are run in protected mode. When protected mode programs are to be started after the RamDisk Server is installed, you must also leave room for their code and some additional room for PMOSS's per-task and per-process system data structures. The best way to determine the exact needs of such programs is to run them, and then trim the RamDisk Server's file list if they do not fit.

PMOSS reclaims the upper memory used for an application when the application finishes, but it can only reclaim such memory in stack order. That is, if application B is started after application A and then A terminates, the upper memory for A will not be reclaimed until B terminates.

Unsupported/Restricted BTOS Interfaces

The following program interfaces to BTOS 8.0 are not supported in protected mode under the current release of PMOSS. This discussion supplements the set of programming rules found in the *BTOS Protected Mode Programming Guide.*

The unsupported operating system interfaces for the current release of PMOSS are:

- The Virtual Code Segment facility, or Overlay Manager (InitOverlays, InitLargeOverlays, ReinitLargeOverlays, ReInitOverlays, GetCParasOvlyZone, MakeRecentlyUsed); programs to be executed in protected mode may not be linked using the /o linker option.
- Mediated interrupt service routines, or any interrupt service routines for other devices than RS-232 serial ports or the Programmable Interrupt Timer (PIT); MediateIntHandler, SetIntHandler, SetTrapHandler, SetLpIsr.
- Raw interrupt service routines for serial ports (comm ports) on the processor module or the XC-002 Port Expander module are supported. Such routines are not permitted to modify the SS register; doing so causes a general protection fault. The stack assigned to a protected mode raw ISR is a dedicated stack, not the stack of some interrupted process, and it is large enough for all reasonable purposes. A protected mode ISR that exceeds this stack will fault. You may want to explicitly change the stack of a raw ISR:
  - To allow more stack space when running the server in real mode. However, even in real mode up to 64 words of stack space should be available to ISRs. Use static rather than local variables in raw ISRs or inner procedures which they call to reduce stack space consumption.

- To make DS and SS the same, so that all code generated by the PL/M compiler in $MEDIUM model will work correctly. If DS addresses DGroup but SS is left unchanged in an ISR, the construct @local, i.e., address of a local variable, will not work, because the compiler assumes DS and SS to be equal when they are not. Carefully purge ISRs and procedures they call of such constructs. This is the only known restriction with using PL/M code in raw ISRs other than performance. Raw comm ISRs should be written in Assembly Language if the comm line is to operate at high baud rates.

- The obsolete RS–232 Comm interfaces (SetCommIsr, SetCommIsrRaw, SetCommIsrRawSfn, ResetCommIsr) are not supported. Use the new interfaces (InitCommLine, ResetCommLine), which are supported (for raw interrupts only).
- The Real Time Clock (RTC) facility is fully supported. Note that CloseRtClock should not be issued while TRB messages may be outstanding (still on an exchange). Drain the exchange first with Check or Wait, or simply don't close and allow termination to clean up for you.
- The Programmable Interval Timer (PIT) facility is supported with the following restrictions:
  - The user-written PIT interrupt service route will be executed with interrupts disabled. It must not enable interrupts during its execution. Because in real mode, PIT interrupt service routines run with interrupts enabled, anyone porting such a routine to run in protected mode under PMOSS should carefully examine it for code which would re-enable interrupts (for example, PL/M code sequences of the form DISABLE; ... ENABLE;).
  - Certain system calls, including the PSend kernel call, are not supported in protected mode PIT interrupt service routines. If you need to wake up a process with a timer, use the Real Time Clock (RTC) facility instead.

- As in real mode, the only permissible system calls in raw comm interrupt service routines are to ReadCommLineStatus, WriteCommLineStatus, SetTimerInt, or ResetTimerInt. However, unlike real mode, PIT interrupt service routines are also restricted to these and only these system calls.
- Parameter management is supported for programs that read parameters (RgParam, CSubParams, CParams) but not for programs that create the Variable-Length Parameter Block VLPB (using RgParamInit).
- GetPStructure supports only the cases listed in the *BTOS Protected Mode Programming Guide*, with the semantics shown there.
- The statusCode = 4 option to GetPartitionStatus (an undocumented option that returns a pointer in real mode) is not supported in protected mode. See SetPStructure in the *BTOS Protected Mode Programming Guide*.
- Multiple run files in the same partition (LoadTask, LoadInteractiveTask) are not supported.
- Certain kernel calls intended for use by the Context Manager or DISTRIX and usually not used by servers or application software are not supported. These include SetDeltaPriority, ChangeProcessPriority, NewProcess, KillProcess, RescheduleProcess, and QuietForSwap. Note that ChangePriority is supported; ChangeProcessPriority, which is not supported is normally used only by system software.
- The ServeSc object module procedure (for installable system common procedures) is not supported.
- SetSegmentAccess is not supported (and is not used by any existing software).
- Send and PSend are fully supported between processes in the same task, but the following restrictions apply when they are used to send messages intertask:
  - The pMsg operand must either have a zero selector or be a pointer to a request block.

- If the latter, the request block must be that of an outstanding request that was originated by the task to which the pMsg is being sent. Originate means sent using the Request or RequestDirect primitive (not ForwardRequest, Respond, Send, or PSend). This restriction applies whenever either or both of the communicating tasks are running in protected mode.
- If the pMsg selector is nonzero, the sender must also place ercInterTaskSend (erc 14106) in the ercRet field of the request block immediately before doing the Send or PSend. This must never be done before an intratask Send or PSend (e.g., a Send between two processes in the same task).
- An intermode Send or PSend operation (one between a real mode and a protected mode task, in either direction) is subject to the above restriction, and an additional restriction if the pMsg selector is nonzero. In this case, the sender must place ercInterTaskSend (erc 14106) in the ercRet field of the request block immediately before doing the Send or PSend. Perform this function before attempting ANY intertask Send or PSend, in both real mode and protected mode servers that communicate using Send or PSend. Do not perform this function before attempting an intratask Send or PSend, for example, a Send between two processes in the same task. Failure to follow these rules will go undetected by PMOSS with unpredictable results.

- The COED facility, which permits reclamation of initialization code memory after initialization, is not supported in protected mode. The current Linker and PMOSS loader will not properly generate code segments for segments of class name COED.

The PMOSS Server itself does not serve a deinstallation request or have a Deinstall command, so PMOSS cannot be deinstalled except by rebooting the workstation. Protected mode servers running under PMOSS can be deinstalled if they support this in real mode, and PMOSS can be enabled and disabled while installed to permit the operator to choose whether to run a program in real mode or protected mode without rebooting.

Servicing an Interrupt

Overhead for servicing an interrupt when the interrupt service routine (ISR) is running in protected mode is greater than when it is running in real mode.

The real mode OS participates in dispatching all ISRs. This is the main reason that protected mode ISRs are at a disadvantage, because a mode switch is always required to dispatch them. In fact, if the processor is already in protected mode when the interrupt occurs, two mode switches are required to dispatch the protected mode ISR.

When an interrupt intended for a real mode ISR (including the BTOS device drivers) occurs while the processor is executing a protected mode program, there is mode switching overhead. There are no noticable latency problems associated with this overhead for the standard BTOS drivers (RS-422 cluster communications, disk, or keyboard.)

Overhead for System Calls

PMOSS also exacts some overhead for mode switching between real and protected mode when kernel calls, system common procedure calls, and requests are made from a protected mode program.

In practice, the servers run in protected mode do not show any visible signs of degrading the system from this increase in overhead. Application programs which are very high frequency users of system common procedures (such as the video output interfaces) might be expected to show signs of performance impact.

Interrupt Latency

Latency is the time it takes to respond to an interrupt (as opposed to overhead, the time it takes to perform the work of servicing the interrupt). Latency problems are caused by programs which keep interrupts disabled for too long a time, and manifest themselves as overrun or underrun errors reported by the communications controller chip when the ISR fails to run soon enough. Refer to the BTOS Status Codes Reference Manual for more information.

Latency problems can occur when the processor utilization is moderate. To solve this problem, break long disabled periods into multiple, shorter disabled periods. It is generally unimportant how long interrupts remain enabled (even one instruction cycle is enough) as long as they do not remain disabled for too long at a stretch.

Latency problems with PMOSS are fairly rare, but have known to occur when an interrupt service routine is inefficient or when a protected mode process performs system calls very frequently.

Most system calls cause a mode switch to real mode and back under PMOSS, during part of which interrupts are disabled. Therefore, a tight loop which performs a system call can cause ISRs to be delayed much more often than usual, causing the controller chip to lose patience and overrun or underrun. Underruns which occur only with synchronous protocols and not with Comm Byte Streams are more likely since the chip has greater patience when receiving than when tansmitting.

ReadCommLineStatus does not do a mode switch to real mode and back, because the practice of polling a status line caused latency problems for some synchronous communications servers. ReadCommLineStatus adds no interrupt latency, because it does not disable interrupts.

Loader Speed

Protected mode programs load more slowly than real mode programs, because the loader in PMOSS reads them from disk one sector at a time.

Request Block Length

Request blocks may not exceed 128 bytes in length. Previous versions of PMOSS restricted request blocks to 64 bytes, which correspond to the limitation imposed by the RS-422 cluster.

A limitation is necessary in PMOSS because like the cluster, PMOSS copies request blocks into small, fixed-length buffers called message buffers. In the case of PMOSS, only the request block is copied, and not the objects pointed to by pb/cbs. The maximum size of a request block affects the storage needed for message buffers, as does the parameter you supplied when installing protected mode.

This restriction applies to all request blocks issued by or sent to protected mode programs.

Error Codes

PMOSS issues error codes in the range 14000 – 14999. The presence of PMOSS is intended to be transparent to the user program and many of these errors reflect an internal problem or configuration error.

The error codes listed in Table A-1 are grouped according to the resource manager which issues them.

Table A-1 **PMOSS Error Codes**

PMOSS – Main Server

Error Code	Meaning
14000	Internal error. (Should never occur.)
14001	Internal error. Indicates an unsupported version of BTOS. Should never be issued, since erc 14025 or erc 14026 is issued first.
14002	The PMOSS server received a request block containing an unexpected request code. Some other server or client may be malfunctioning.
14003	Internal error. (Should never occur.)
14004	Internal error. (Should never occur.)
14005	Internal error. (Should never occur.)

PmAgent - Real Mode Agent for Main Server

Error Code	Meaning
14006	Like any server, PMOSS cannot be installed after the Context Manager is installed. Deinstall Context Manager and try again.
14007	Too little RAM is installed. Add RAM expansion card(s).
14024	Real mode processor. This CPU supports only real mode; PMOSS cannot be installed. Intel 80286 and 80386 microprocessors support protected mode.
14025	Down—rev BTOS. BTOS version 8.0 or later is required to install PMOSS.
14026	Single—partition BTOS. PMOSS requires multipartition BTOS.
14027	PMOSS already installed.
14028	SPA mover server (Mover.run) already installed. Install either PMOSS or the Mover.run server, but never both.
14029	PMOSS resident debugger inactive. An internal error, should never occur.
14060	No mover handle available. This error occurs when the [Maximum number of mover server segments] parameter is exceeded. Reboot and reinstall PMOSS, specifying a larger value. Note that the maximum value of this parameter is 2048. The size of the real mode PmAgent.run partition will not increase, but the amount of available upper memory will decrease slightly.

Table A-1 **PMOSS Error Codes (continued)**

PMOSS Mover Server (serves SPA Mover requests)

Error Code	Meaning
14061	Bad mover address. Unlike the SPA Mover.run server which PMOSS replaces, PMOSS validates requests to move to or from protected memory. The specified range of bytes must lie entirely within a single mover segment that was previously allocated by the client (that is, the userNum field in the move request must match the userNum in the allocation request). Move requests may not span mover segments, even when both were allocated to the same userNum.
14062	Mover address buffer too small. The cbRet field of a mover allocation request is less than 4 bytes long, so PMOSS cannot return the 4 byte segment identifier.
14063	Mover version buffer too small. The cbRet field of a mover version request is less than the required size (2 bytes).

Kernel Interface Manager

Error Code	Meaning
14100	Out of message buffers. This error occurs when the [Maximum number of concurrent protected mode messages] parameter is too small. This line of the command form accepts two values (maximum number of small and large messages, respectively). Reboot and reinstall PMOSS, specifying larger values. The size of the real mode PmAgent.run partition will increase. This error may indicate either (a) all message buffers are full, or (b) all large message buffers are full and a large message buffer is required. PMOSS will use large message buffers for small messages if all small message buffers are full.
14102	Message too large. An invalid request block was encountered. The request block appeared to be larger than 64 bytes long, perhaps because the nReqPbCb and nRespPbCb fields in its header were garbage. If this error is returned from Request, Respond, or ForwardRequest, the protected mode caller provided an invalid request block. If it is returned from Wait, the invalid request block was received from a real mode program, or an internal error occurred in PMOSS.
14104	Reserved.
14105	Reserved.

Table A-1 **PMOSS Error Codes (continued)**

Kernel Interface Manager

Error Code	Meaning
T214106	Intertask Send. The user must mark request blocks sent between tasks using Send or PSend with this value in the ercRet field, whenever either sender or receiver (or both) is a protected mode task. This erc does not indicate an error condition.

Selector Manager (manages descriptor tables)

14120	Global Descriptor Table (GDT) full. Internal error. Rebuild PMOSS from source with a larger GDT specified in the PMake step.
14121	Local Descriptor Table (LDT) full. This may indicate a user programming error, such as the allocation of an unusually large number of very small memory segments, or an internal error in PMOSS. PMOSS does not support automatic expansion of a client program's LDT when it overflows.
14122	Invalid selector. An invalid selector was used in an operating system call, such as a memory deallocation request. (More frequently, use of an invalid selector results in a General Protection Fault.) Since selectors become invalid when the objects they point to are deallocated, an attempt may have been made to free the same memory twice, for example.

Virtual Segment Manager (PMOSS memory management)

Error Code	Meaning
14140	Invalid qb (size) parameter. This erc would not normally be returned to a user program that called a BTOS interface, and indicates an internal PMOSS error.
14141	Nonsegment selector. The selector specified for this operation was expected to identify a segment descriptor, but it does not. It may identify some other form of descriptor, such as a call gate. Similar to erc 14122.
14143	No such virtual segment. Similar to 14122.
14144	Not paragraph aligned. An attempt was made to pass a protected mode pointer or selector in a request, system common procedure, or kernel call which is not translatable to a real mode address because it refers to a segment which is not paragraph aligned. See error code 14145.

Table A-1 **PMOSS Error Codes (continued)**

Virtual Segment Manager (PMOSS) memory management (continued)

Error Code	Meaning
14145	Not accessible from real mode. An attempt was made to pass a protected mode pointer or selector in a request, system common procedure, or kernel call which is not translatable to a real mode address because it refers to a segment which is not in real mode memory (address range 0992KB). The usual cause of this error is passing a pointer to data contained in a code segment. This is not permitted in protected mode because code segments reside in the upper megabytes. Note that even when the recipient of the request is another protected mode program, pb/cb pointers must be translated to real mode addresses and back en route.

Linear Segment Manager (upper memory management)

14160	Not enough memory. Occurs when upper memory is full.
14161	Invalid heap ID. Internal PMOSS error.
14162	LSD already free. Internal PMOSS error.

Active Run File Manager (protected mode loader)

Error Code	Meaning
14180	No such run file. Should not be returned to user programs. Internal PMOSS error.
14181	Bad run file. The run file header is invalid.
14182	Buffer too small. Internal PMOSS error.

Task Manager (LDT and task resource management)

14200	No pending task. An internal error occurred while a protected mode program was loading (specifically, PMOSS received an unexpected request from an instance of PMSubAgent.run.
14201	Invalid default user number in effect for process at ConvertToSys time. Should not occur in PMOSS.
14202	Swap status error. Internal to PMOSS.
14203	Cannot Deinstall PMOSS.

Table A-1 **PMOSS Error Codes (continued)**

Process Manager (TSS management)

Error Code	Meaning
14220	Bad process descriptor (the structure used as an argument to CreateProcess). This is an internal error which should not be seen by user programs.

CommLine Manager (RS-232 serial port manager)

14240	No free CommLine control structure available. Internal error. This should never occur, since PMOSS is built with enough of such control structures for ten serial ports, the maximum possible on a workstation (with two XC-002 Port Expander modules in place).

Get/SetPStructure

14260	Bad structCode operand to GetPStructure. This value of the structCode parameter is not supported.
14261	Bad structCode operand to SetPStructure. This value of the structCode parameter is not supported.
14262	Bad SetPStructure field offset. This particular field is not supported.
14263	Reserved.
14264	Count of bytes operand (cb) too large for SetPStructure. The operand exceeds the maximum allowable size for this field, which is an sbString (a string preceded by a length byte). When setting such fields with SetPStructure, the pb and cb operands should not include the length byte; rather, the length byte will be set to the value of cb automatically.
14265	Unexpected count of bytes operand (cb) for SetPStructure. For a fixedlength field, the operand's value did not exactly match the size of this field.
14266	The task attempted to perform the GetPStructure operation for too many different data structures. PMOSS has a limit of 64 distinct OS data structures that a task may know about. Note that you may call GetPStructure as many times as desired for the same OS data structure without producing this error.

Table A-1 **PMOSS Error Codes (continued)**

ASCII String Manager

An explanatory message, rather than any of these error codes should be issued by PmAgent if the command form is incorrectly filled in. The operator should not normally see there ercs.

Error Code	Meaning
14280	Invalid decimal integer.
14281	Overflow. A decimal integer exceeds DWORD precision.
14282	Null string. Not necessarily an error, but an indication of a missing parameter.
14283	Neither yes nor no. An invalid yes/no string.
14284	Hex result field too small. A procedure that returns a hexadecimal ASCII string requires a larger sStringMax parameter. The result string was truncated on the right.

Glossary

Descriptors. Descriptors are eight bytes long and contain various information about a segment. They are basically an offset into the Logical Descriptor Table, with some additional bits used for special purposes.

Descriptor Table. A descriptor table is a special type of table that only the operating system and hardware can access. In protected mode, the CS, DS, ES, and SS registers hold a 16-bit SA that is an index into the descriptor table.

External Interrupt. An external interrupt is initiated by an asynchronous event outside the processor. External interrupts are I/O interrupts such as disk and real-time clock interrupts.

Fault. A fault is a condition (such as a protection violation) that prevents the hardware from completing an instruction.

Gate Descriptor. A gate descriptor is a structure that uses indirection to allow programs to call routines whose addresses they cannot know until the program is loaded.

General Protection Faults. General protection faults are generated by breaking any of the protection rules, which activate the Debugger.

Global Descriptor Table (GDT). The Global Descriptor Table is a special table that is PMOSS' LDT and the descriptors in it are used only when PMOSS' code is executing, never when user code is executing. The single Global Descriptor Table is never switched; it is always in effect, no matter what Local Descriptor Table is in effect.

Internal Interrupt. An internal interrupt is initiated synchronously as a result of an instruction executing. Internal interrupts include software interrupts, (which happen when an INT instruction executes), exceptions (such as interrupt type 4), and faults (including protection faults).

Interrupt Descriptor Table (IDT). An Interrupt Descriptor Table is a special descriptor table that contains the interrupt type numbers corresponding to every interrupt. There is one Interrupt Descriptor Table per system.

Linear Address. A linear address is formed from the logical address as an instruction executes and then addresses physical memory.

Linear Address Model. A linear address model refers to an architecture (such as the Motorola architecture) in which instructions accept 32-bit linear addresses (instead of SA:RA pairs, as with the segmented addressing model).

Logical Address. A logical address is composed of the segment address (SA) and the relative address (RA). You use the SA:RA syntax to write a logical address when using the Debugger or Assembler.

Local Descriptor Table (LDT). The Logical Descriptor Table is an array of descriptors. PMOSS constructs and maintains the Logical Descriptor Table for each run file executing in protected mode.

Paragraph. A paragraph is a 16-byte unit of memory aligned on a 16-byte boundary.

Paragraph Number. A paragraph number is a real address mode SR, which denotes a particular 16-byte boundary in the physical address space.

Protected Mode Operating System Server (PMOSS). Protected Mode Operating System Server installs on BTOS and lets you write programs that are compatible with protected mode (the 3 Mb over the 1 Mb of real mode).

Real Address. The real address is the real address mode SR:RA logical address because it always corresponds to the same physical memory address.

Real Address Mode. Real address mode is the mode in which 8086 and 80186 microprocessors operate all the time. 80286 microprocessors operate in real address mode when powered-up or reset, but can switch to protected mode if the operating system software supports protected mode.

Relative Address (RA). The relative address comprises one-half of the linear address. It is often referred to as an offset from the segment address.

Restartable Fault. A restartable fault is a fault (such as a not-present fault) that is, in theory, recoverable.

Segment Address (SA). The segment address comprises one-half of the linear address.

Segmented Address Model. A segmented address model refers to the Intel architecture in which every address is always relative to some segment address (SA).

Segment Registers. Segment registers contain the paragraph numbers corresponding to the vase of the current code (CS), data (DS), extra (ES), and stack segments (SS). These segments are always aligned to start on 16-byte boundaries.

SN. The SN is a segment address that is in protected mode.

SR. The SR is a segment address that is a paragraph number (a real-address mode SA).

System Performance Accelerator (SPA). The System Performance Accelerator is an installed system service that improves the response time for workstations performing file-system operations and also provides a caching mechanism.

Task State Segment (TSS). The Task State Segment is associated with every process and contains the complete register state of the process. It permits extremely rapid process switching, despite protection boundaries between programs.

TSS Gate. A TSS gate lets you arrange for an interrupt to perform a process switch automatically (in effect, an automatic Call to a TSS).

Virtual Address. The virtual address is the protected mode SN:RA logical address because the SN refers only indirectly to memory via a descriptor table entry.

Index

I

L

M

O

P

Title: _____

Form Number: _____ Date: _____

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information.

Please check type of suggestion: ☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments: _____

_____

_____

_____

Name _____

Title _____

Company _____

Address _____
Street City State Zip

Telephone Number ( ) _____
Area Code

Title: _____

Form Number: _____ Date: _____

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information.

Please check type of suggestion: ☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments: _____

_____

_____

_____

Name _____

Title _____

Company _____

Address _____
Street City State Zip

Telephone Number ( ) _____
Area Code

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

POSTAGE WILL BE PAID BY ADDRESSEE

Unisys Corporation
1300 John Reed Court
City of Industry, CA 91745 USA

ATTN: Corporate Product Information

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

POSTAGE WILL BE PAID BY ADDRESSEE

Unisys Corporation
1300 John Reed Court
City of Industry, CA 91745 USA

ATTN: Corporate Product Information

Burroughs