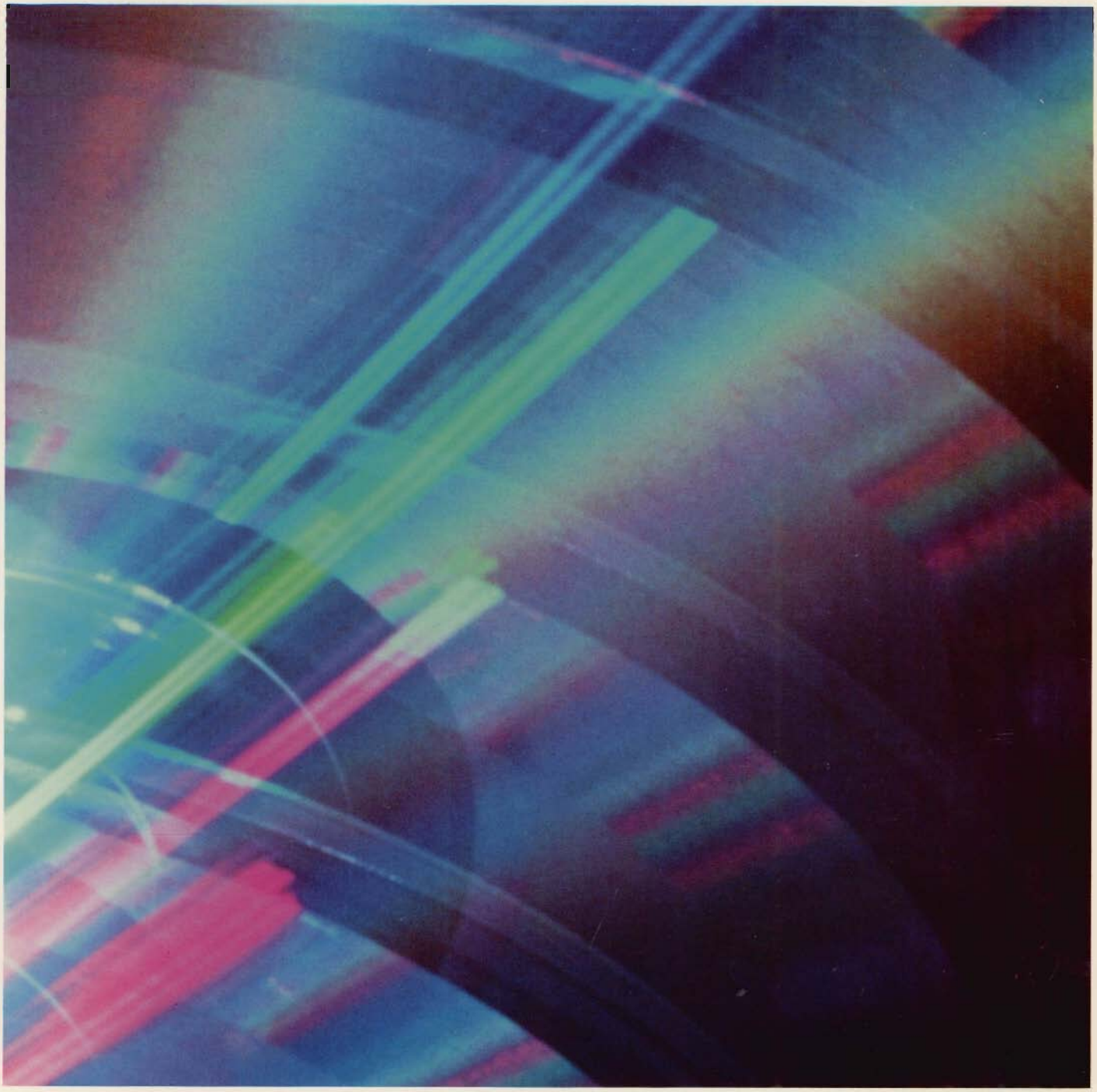
HEWLETT PACKARD

MPE XL DAT Macro

Reference Manual



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

READER COMMENT SHEET

Information Technology Group

MPE XL DAT Macro Reference Manual

Manual Part Number 32650-90094 December 1987

A reader comment sheet helps us to improve the readability and accuracy of the document. It is also a vehicle for recommending enhancements to the product or manual. Please use it to suggest improvements.

SERIOUS ERRORS, such as technical inaccuracies that may render a program or a hardware device inoperative should be reported to your HP Response Center or directly to a Support Engineer. An engineer will enter the problem on HP's STARS (Software Tracking and Reporting System). This will ensure that critical and serious problems receive appropriate attention as soon as possible.

Editorial suggestions (please include page numbers):_____

_____

_____

_____

Recommended improvements (attach additional information, if needed):_____

_____ _____

_____

_____

Name _____ Date _____

Job Title _____ Phone_____

Company_____

Address _____

HP 1000 Series_____ (e.g., E-series, A400, A600, etc.)
HP 3000 Series_____ (e.g., 37, 68, 930, etc.)
HP 9000 Series_____ (e.g., 300, 840, etc.)

☐ Check here if you would like a reply.

Hewlett-Packard has the right to use submitted suggestions without obligation, with all such ideas becoming property of Hewlett-Packard.

Fold Here

Fold Here

NO POSTAGE NECESSARY IF MAILED IN THE UNITED SATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070, CUPERTINO, CA 95014

- POSTAGE WILL BE PAID BY -

Publications Manager
Hewlett-Packard Company
Information Technology Group - Hardware Documentation
19483 Pruneridge Avenue
Cupertino, California 95014-9974

Fold Here

Fold Here

READER COMMENT SHEET

Information Technology Group

MPE XL DAT Macro Reference Manual

Manual Part Number 32650-90094 December 1987

A reader comment sheet helps us to improve the readability and accuracy of the document. It is also a vehicle for recommending enhancements to the product or manual. Please use it to suggest improvements.

SERIOUS ERRORS, such as technical inaccuracies that may render a program or a hardware device inoperative should be reported to your HP Response Center or directly to a Support Engineer. An engineer will enter the problem on HP's STARS (Software Tracking and Reporting System). This will ensure that critical and serious problems receive appropriate attention as soon as possible.

Editorial suggestions (please include page numbers):_____

_____

_____

_____

Recommended improvements (attach additional information, if needed): _____

_____ _____

_____

_____

Name _____ Date _____

Job Title _____ Phone_____

Company_____

Address _____

HP 1000 Series_____ (e.g., E-series, A400, A600, etc.)
HP 3000 Series_____ (e.g., 37, 68, 930, etc.)
HP 9000 Series_____ (e.g., 300, 840, etc.)

☐ Check here if you would like a reply.

Hewlett-Packard has the right to use submitted suggestions without obligation, with all such ideas becoming property of Hewlett-Packard.

Fold Here

Fold Here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED SATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070, CUPERTINO, CA 95014

- POSTAGE WILL BE PAID BY -

Publications Manager
Hewlett-Packard Company
Information Technology Group - Hardware Documentation
19483 Pruneridge Avenue
Cupertino, California 95014-9974

Fold Here

Fold Here

READER COMMENT SHEET

Information Technology Group

MPE XL DAT Macro Reference Manual

Manual Part Number 32650-90094 December 1987

A reader comment sheet helps us to improve the readability and accuracy of the document. It is also a vehicle for recommending enhancements to the product or manual. Please use it to suggest improvements.

SERIOUS ERRORS, such as technical inaccuracies that may render a program or a hardware device inoperative should be reported to your HP Response Center or directly to a Support Engineer. An engineer will enter the problem on HP's STARS (Software Tracking and Reporting System). This will ensure that critical and serious problems receive appropriate attention as soon as possible.

Editorial suggestions (please include page numbers):_____

_____

_____

_____

Recommended improvements (attach additional information, if needed): _____

_____ _____

_____

_____

Name _____ Date _____

Job Title _____ Phone_____

Company_____

Address _____

HP 1000 Series_____ (e.g., E-series, A400, A600, etc.)
HP 3000 Series_____ (e.g., 37, 68, 930, etc.)
HP 9000 Series_____ (e.g., 300, 840, etc.)

☐ Check here if you would like a reply.

Hewlett-Packard has the right to use submitted suggestions without obligation, with all such ideas becoming property of Hewlett-Packard.

Fold Here

Fold Here

NO POSTAGE NECESSARY IF MAILED IN THE UNITED SATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070, CUPERTINO, CA 95014

- POSTAGE WILL BE PAID BY -

Publications Manager
Hewlett-Packard Company
Information Technology Group - Hardware Documentation
19483 Pruneridge Avenue
Cupertino, California 95014-9974

Fold Here

Fold Here

900 Series HP 3000 Computer Systems

MPE XL DAT Macro

Reference Manual

HEWLETT PACKARD

Edition 1 E1287

32650-90094
Printed in U.S.A. 12/87

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

First Edition ..... December 1987

List of Effective Pages

The List of Effective Pages gives the date of the most recent version of each page of the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Page	Date
All	December 1987

Safety Considerations

This product and related documentation must be reviewed for familiarization with safety markings and instructions before operation.

WARNING

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

CAUTION

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

Table of Contents

Preface

This manual contains information about the Dump Analysis Tool (DAT) macros, which are used to troubleshoot 900 Series HP 3000 Computers. It is intended to be used as technical support hardware documentation for Hewlett-Packard SEs and other qualified support personnel. The procedures and software focus primarily on the hardware troubleshooting environment of MPE XL and require specific training for correct and safe usage.

Specifically, this manual describes the DAT macros. These provide an orderly method for investigating system failures in the MPE XL operating system environment.

Available DAT Macros

Current State Macros

file_in_use	Prints all files that are currently open for each PIN.
global_memory	Prints information about global memory.
job_oper_req	Prints any outstanding operator job requests.
machine_state	Prints the state of the processor plus general dump information.
process_dispatcher	Prints information about the current state of the dispatcher.
process_wait	Prints all waiting processes.
res_all	Prints all high-level information on system resources.
res_crit_held	Lists held critical resources by resource type.
res_crit_waiting	Lists critical resources that are being waited on by a process.
res_deadlock	Prints all deadlocked resources by type.

All Other Macros In Alphabetical Order

config_device_ldev	Prints configuration information on one or a -al devices.
config_device_path	Prints information for every I/O path on the system.
config_memory	Prints the physical memory configuration information.
console_ldev	Prints the logical device number for the system.
event_ci_history	Prints all CI redo stacks for all live processes.
event_footprint	Prints the global footprint information.
event_io_trace	Prints IO monitor table information.
event_process	Prints the local process footprint table.
event_process_errors	Prints process error stacks.
fs_access_attributes	Prints the attributes of the specified file.
fs_acc_attr	Prints the attributes of the specified file.
fs_acc_tree	Prints an access tree for a NM shared file.
fs_addr	Prints the virtual address and length of the specified file.
fs_address	Prints the virtual address and length of the specified file.
fs_aft	Returns the pointer to the CM AFT entry corresponding to the specified PLFD pointer.
fs_aftcm	Returns the pointer to the CM AFT entry corresponding to the specified CM stack and file number.
fs_all_files	Prints all information about files for a given process.
fs_aoptions	Returns the access options (aoptions) for the PLFD entry.
fs_cmcb	Returns the virtual address of the CM FILE CONTROL BLOCK given the CM data segment from the AFT.
fs_cmcbt	Returns the virtual address of the CM FILE CONTROL BLOCK TABLE given a pointer to the data segment.
fs_cmstack	Returns the pointer to the CM stack which contains the AFT for the file corresponding to the specified PLFD.
fs_dev	Prints logical device identification corresponding to the file.
fs_device	Prints the logical device id for the specified file.
fs_disc_alc	Prints the disc allocation map for the specified file.

fs_disc_alloc	Prints the disc allocation map for the specified file.
fs_fcb	Returns the pointer to the CM FCB entry corresponding to the specified PACB pointer.
fs_file	Prints all information about a file.
fs_filename	Prints the filename of the specified file.
fs_filename_ufid	Prints the filename of the file for the specified ufid.
fs_file_perm	Prints all information about a permanent file.
fs_file_ufid	Prints all information about a file.
fs_flab	Returns the file label pointer for the GUFD.
fs_fname	Returns the file name that corresponds to the PLFD.
fs_fname_to_gufd	Returns the address of the GUFD for the specified filename.
fs_foptions	Returns the access options for the specified PLFD entry.
fs_format_aoptions	Returns the specified aoptions formatted in a string.
fs_format_foptions	Returns the specified foptions in a string.
fs_gufd	Returns the address of the GUFD based on the PLFD address.
fs_lacb	Returns the pointer to the CM LACB entry corresponding to the specified PLFD pointer.
fs_mounted_volumes	Prints and returns pointers to the Mounted Volume Table entries of all volumes mounted on the system.
fs_mvt	Returns the address of the Mounted Volume Table entry for the specified GUFD.
fs_next_gdpd	Returns the address of the next GDPD for the specified GUFD.
fs_pacb	Returns a pointer to the CM PACB entry which corresponds to the specified PLFD pointer.
fs_perm_char	Prints the permanent characteristics for the specified file.
fs_plfd	Returns the address of the PLFD based on pin and file number.
fs_pxfile	Returns a pointer to the PXFILE area of the CM stack given a pointer to the beginning of the stack.
fs_scan_fmavt	Returns a pointer to the FMAVT entry corresponds to the specified GUFD pointer.
fs_ufid	Prints the UFID of the specified file.
fs_ufid_file	Prints the UFID associated with the filename.
fs_ufid_str	Returns the UFID associated with the GUFD.
fs_ufid_to_gufd	Returns the GUFD associated with the specified UFID.
fs_vol_name	Returns the volume name for the specified Mounted Volume Table entry.
fs_vtab_entry	Returns the pointer to the CM file Control Block Table Vector table entry given the CM segment-offset vector from the AFT.
io_cm_ioreq	Prints the CM I/O request entry associated with the specified IOX.
io_config_ldev	Prints device configuration information for the specified logical device.
io_config_path	Prints device configuration information for the specified path specification.
io_data_chain	Print or returns the specified field form the data chain record.
io_dct	Print the Device Class Table (DCT) entry associated with the specified device class.
io_docket	Prints the docket table for the IOLDM for the specified LDEV.
io_hlio_global	Print HLIO Global area (SANCTUM).
io_ioldm_port	Prints the port data area for the IO/LDM port for the specified LDEV.
io_ldev_owner	Returns the pin that owns the specified ldev.
io_ldev_to_path	Returns the path specification corresponding to the specified logical device.
io_ldt	Print the CM LDT entry for the specified LDEV.
io_lpdt	Prints the CM LPDT entry for the specified LDEV.
io_lpt	Prints the LPT entry for the specified LDEV.

io_mib	Prints the specified MIB entry.
io_path_to_ldev	Returns the logical device number corresponding to the specified path name.
io_portid	Returns the portid associated with the specified LLIO path.
io_timer_list	Formats the timer request list.
pm_devices	Prints or return a list of devices owned by the specified process.
pm_disp_global	Prints the specified information in the dispatcher global area.
pm_disp_queue	Prints the dispatcher dispatch queue.
pm_fpib	Formats pib entries for all processes.
pm_fpibx	Formats pibx entries for all processes.
pm_ics_header	Prints the specified information in the ICS header area.
pm_kpo_number	Returns the number of the specified Know Process Object (KPO) constant.
pm_kpo_pointer	Returns the virtual address of the specified Known Process Object (KPO) for a specified job.
pm_loaded_file_table	Prints the specified information in the Loaded File Table for the specified UFID.
pm_pibx_info	Prints or returns the specified field from the pibx for the given pin.
pm_pib_info	Prints or returns the specified field from the pib for the given pin. Will format the entire pib or just return a field.
pm_ptree	Prints a process tree for a specified process, and returns a list of pins printed.
pm_scheduling	Prints the scheduling information associated with the specified process.
pm_semaphores	Prints virtual address of semaphore being waited on, if any.
rm_format_sirs	Formats the compatability mode SIR table and also checks for deadlocks.
rm_global_deadlock	Checks all PINs for semaphore and SIR deadlock and returns list of suspect PINs.
rm_semaphore_info	Prints or returns the type, class, and semaphore specific information for the specified semaphore address.
rm_sem_blocked_proc	Prints or returns a list of pins that are blocked on the specified semaphore.
rm_sem_deadlock	Checks all PINs for semaphore deadlock and returns list of suspect PINs.
rm_sem_owner	Prints or returns the owner of the specified semaphore.
rm_sir_blocked_proc	Prints the sirs owned and being waited on form the compatability mode sir table.
rm_sir_deadlock	Checks all PINs for SIR deadlock and returns list of suspect PINs.
rm_sir_owner	Prints or returns the owner of the specified SIR.
ui_capabilities	Prints the capabilities information associated with the job.
ui_ccb	Prints the specified CCB table or field for the specified CI process.
ui_cicommand	Prints the command interpreter variables for the job.
ui_cihistory	Prints the Command Interpreter history stack of the job.
ui_ciprocess	Prints the list of command interpreter processes for the specified job.
ui_civariables	Prints the command interpreter variables associated with the job.
ui_devices	Prints and returns the devices associated with the job.
ui_jdt	Prints the JDT entry associated with the specified job.
ui_jit	Prints the JIT associated with the specified job.
ui_jmat	Prints the JMAT entry associated with the specified job.
ui_job	Prints all information about a job.
ui_jpcnt	Returns the JPCNT number associated with the specified job.
ui_process	Prints the process tree associated with the job number.
ui_reply	Prints a report of all replies pending.
ui_schedule	Prints the scheduling information associated with the specified job.
ui_tempfiles	Prints any temporary files or file equation associated with job.
ui_uit	Prints the UIT for the specified pin.
ui_user	Prints the full user/account name associated with the job.

Conventions

NOTATION	DESCRIPTION
UPPERCASE	Within syntax statements, characters in uppercase must be entered in exactly the order shown, though you can enter them in either uppercase or lowercase. For example: `SHOWJOB` Valid entries: `showjob` `ShowJob` `SHOWJOB` Invalid entries: `shojwob` `Shojob` `SHOW_JOB`
italics	Within syntax statements, a word in italics represents a formal parameter or argument that you must replace with an actual value. In the following example, you must replace *filename* with the name of the file you want to release: `RELEASE` *filename*
punctuation	Within syntax statements, punctuation characters (other than brackets, braces, vertical parallel lines, and ellipses) must be entered exactly as shown.
[ ]	Within syntax statements, brackets enclose optional elements. In the following example, brackets around `,TEMP` indicate that the parameter and its delimiter are optional: `PURGE` *filename*`[,TEMP]` When several elements within brackets are stacked, you can select any one of the elements or none. In the following example, you can select *devicename* or *deviceclass* or neither: `SHOWDEV` [*devicename*] [*deviceclass*]
`shading`	Within an example of interactive dialog, `shaded` characters indicate user input or responses to prompts. In the following example, `OMEGA` is the user's response to the NEW NAME prompt: NEW NAME? `OMEGA`

Conventions (Continued)

NOTATION	DESCRIPTION
[ ]	The symbol [ ] indicates a key on the terminal's keyboard. For example, [CTRL] indicates the Control key.
[CTRL] *char*	[CTRL] *char* indicates a control character. For example, [CTRL] Y means you have to simultaneously press the Control key and the Y key on the keyboard.
base prefixes	The prefixes %, #, and $ specify the numerical base of the value that follows: *%num* specifies an octal number *#num* specifies a decimal number *$num* specifies a hexadecimal number When no base is specified, decimal is assumed.

Contents

Chapter 1
Overview of DAT Macros

1

Overview of DAT Macros

Introduction

The Dump Analysis Tool (DAT) macros will be used by both field and lab personnel to resolve problems with system software. The macros will be specified so that both experts in the software and non-experts will be able to effectively look at the status of a dump.

The macros are designed primarily for interactive use, allowing the intelligent, formatted display of specific, small amounts of table information. There are also macros which produce a large quantity of formatted table information. These can be used to prepare printed dumps.

General Parameters

Usage Requirements

In order to use the DAT macros, one must have available on the system the proper groups and files in the TELESUP account to support DAT. These include:

- An 8.6 or later version of DAT.
- A Dump file (readable by DAT 8.6 or later). If the file is on tape, the TELESUP environment will allow you to read it onto disc once inside DAT.
- The DAT MACROs and matching SOM file(s).

The macros must be accessed from inside DAT. The accounting structure has recently been changed to allow you to load the macro environment without memorizing specific DAT commands. One should now merely log into the TELESUP account in the DAT group, then type DAT to run the program DAT.DAT.TELESUP.

This program will now automatically use an initialization file in that group called DATINIT which does the following things: Prompts the user to see if a dump file needs to be loaded, prompts the user for the dump file set name if the dump is already loaded, and opens the dump file set. It then will go to the group needed to find files which are OS version-dependent. Its will use these files to unwind the dump, load the SOM files, set up some initial variables, and load the macros themselves.

Although it is possible to USE all the macro files individually without using DATINIT, this is not suggested as many dependencies exist. The user will be notified if anything fails. Within a DAT session, multiple dumps can be accessed with or without reusing DATINIT.

Objectives

The macros are being specified and designed with the following objectives:

Operating System Invariant
The macros are structured in layers such that the top levels appear to the user as independent of the specified operating system they are implemented on. This will mandate a minimum of retraining when other operating systems are implemented in conjuction with MPE XL. Where the user interface must be operating system specific, this will be clearly isolated and documented.

Non-Expert Interface
The top two levels of the macros will be designed for the "non-expert" user. This is a user who is not trained in MPE XL internals, but is trained in the maintenance of a multiprogramming operating system.

Maximum Leverage of Macro Code
All macros will be constructed to return values such that they can easily be used in combination with other macros to provide the maximum amount of re-use of macro code.

Minimize Maintenance Effort
The operating system dependent code will be as isolated as possible to minimize the impact of operating system changes. Futher, symbolic formatting will be used whereever possible.

Organization/Content of Macro Descriptions

This macros are structured around diagnostic approaches to a dump. The macros provided are designed for non-experts to use and are operating system invariant both in the user interface and the internal structure. These macros will (for the most part) only call the low-level and general macros. Since dumps may be approached in different, overlapping ways, these macros will be quite redundant in some areas; this will facilitate the analysis of the dump by non-expert users.

Functionality
Each section will contain a major area to look at in a dump (e.g. current status, recent events).

Syntax
The syntax, described using the conventions at the beginning of this manual. Also included in the syntax specification is the type of each argument (shown after the colon). This type is the actual type defined in the macro definition for this parameter.

Parameters
A description of each of the parameters listed in the syntax and a notation that the parameter is optional or required.

User Discussion
Conditionally, based upon the printer's desires, a discussion of how the macro works internally, and a list of all macros used by this macro are included. This is used to understand the exact method the macro uses to find the table so that problems can be resolved if the macro fails. It is also used to determine if maintenance needs to be done on the macro as tables change.

Example
An example of some uses of the macro.

Data Structure Formatting

The formatting of data structures in the expert level macros will be accomplished through the formatting function of the DAT/Debugger whenever possible. The formatting for the non-expert macros will be handled explicitly within the macro.

It is strongly recommended that the expert reader become very familiar with the symbolic formatting section of the DAT/Debugger manual. This function allows the formatting of a given data area according to a MODCAL Type definition found in a System Object Module (SOM) file which is specified in advance.

For the SPL compatability mode software, MODCAL type definitions will be prepared by a manual conversion of the SPL record definitions.

Maintenance

This section provides information necessary to maintain the macros. Due to the use of symbolic formatting, the general rule is macros will not need to be changed unless tables are deleted, tables are added, or the linkages between existing tables are changed. The exceptions to this are macros which use SYSGLOB cells as pointers, and use stack DB relative values to point to tables. Whenever these pointers are changed, the macros will need to be changed. Macros will not need to be changed if table sizes are changed, new items are added to tables that do not affect the existing linkages, and items are deleted from tables that do not affect the existing linkages. Each macro will contain a list of the called macros.

In addition to maintaining the macros, the MODCAL type definitions which correspond to the compatability mode SPL subsystems need to always be kept in sync with their SPL counterparts. These include type and constant definitions. The names of these files are as follows:

TCMFS.DATSYM.OFFICIAL -MPE V File System definitions.

Standards

The following guidelines will apply in the design of all macros:

Returning Values

There are a limited set of standards which must be adhered to to achive consistency among the macros.

Coding Standards

All local variables used in a macro are to be defined at the beginning of the macro and explicitly typed. Initialization is permitted in the definition, if practical. Note that since DAT does not allow definition of a variable without setting it to something, the variables not needing initialization must appear as comments (eg: \loc ufid : str;).

The types INT and PTR should be used whenever practical instead of explicitly declaring the type of integer (e.g. U32) or pointer (LPTR).

All output will be formatted to fit within 80 characters and be designed to be used with a terminal.

The macros should be formatted to be presentable and pleasing to review. At the same time, space should be conserved whenever possible, since there is a 2000 character limit on the size of a macro. Leading and trailing spaces do not count against this limitation, since they are stripped by the macro pre-processor.

String literals may not be split across lines. When a long string is necessary, the concatenation feature should be used.

All macros are to be contained within braces ({). The first brace should go immediately before the mac statement, on the same line. The last brace should be a double close brace (}}), terminating the macro, and the brace before the macro.

Macros will not return a special value indicating an error or anything else. The value returned must always be what is expected (whether it is valid or not). The only exception to this is end of list handling. If there is an end of list condition, the following will be returned:

```
0 (NMNIL) - If the type of the return is a pointer.
-1        - If the type of the return is an integer.
```

Whenever a macro encounters an error condition, the value returned will be considered unpredictable.

When DAT functions return errors, they should be left to blow up completely (return back to the user). In cases where execution should continue, the DAT ignore statement should be used immediately before the DAT function or command. Even though execution will continue in the event of an error, an error message will still be printed out by DAT. Also, a syntax error will always cause execution to abort. The IGNORE DAT command should never be used for anything else.

To access the Pascal/XL (MODCAL) value NIL use NMNIL. This will be a DAT global variable (constant) defined when the macros are used. Note that when the need arises to have a long pointer NIL value, the variable NMNILL can be used.

Macro File Naming

Presently, the TELESUP account is being used to house DAT and the files needed to set up the macro environment. This includes DAT.DAT.TELESUP, and several other files in the DAT group (including DATINIT). Once the DATINIT use file has figured out what version of the OS was running at the time the dump was taken, it goes to specific groups within the TELESUP account to unwind the dump, set up the basic macro environment, and load the macros. The present convention is to have these groups be OSxxx, where xxx is taken from the OS vuf (ie: 9.00.zz files will reside in the OS900.TELESUP group). These OSxxx groups then contain use files, a symbolic file (SYMOS), and macro use files.

For non-macro USE files, the naming convention is that they begin with U, and are fairly mnemonic. For example, and files used to unwind, set up the environment, and load the macros are called UUNWIND, USETUP, and UMLOAD.

For macro use files, the following naming convention will apply:

Mxyy - where:

M	indicates a macro use file.
x	is the level (eg: H=high, L=low, X=expert).
yy	is the subsystem code (eg: FS, IO, PM, ect.).

The general macro file is an exception to this rule. Since it has no subsystem code (cuts across all subsystems), it will be called MGEN.

Normal macro users should not need to specifically USE any file, as this should be covered by DATINIT, and the files it uses. The NS and other subsystem macros may adopt this same naming convention. Currently, DATINIT uses DCINIT and DBINIT, which also reside in the DAT group, to initialize the Datacomm and Database macro environments.

Macro Naming

Macro names will be in the following format:

`ss_xxx...xxx` - where:

ss	subsystem code. (eg: fs,io,pm,etc.)
xxx...xxx	anything which describes the macro

The high level and general macros will be exceptions to this rule in that they will not have a subsystem code. Also, commonly used macros may have shortened "aliases" to reduce typing fatigue. The alias macros will merely be made up of a call to the longer-named macro it is aliasing.

SOM Files

Since SOM files are used for symbolic formatting, the way the macros refer to the SOM files should be through global variables, rather than have the filenames wired into the macros. This will allow a single development SOM file to be used in conjuction with the standard production SOM files.

A list of all of the SOM file identifiers is included here. Although there will normally be only one SOM for the entire operating system (SYMOS), subsystem specified SOMs can be substituted by changing the value of the variables listed below.

Compatability mode OS	cm_som
Compatability mode ports	pc_som
File system	fs_som
HPE definitions	hp_som
High level input/output	hi_som
Job management/command interpreter	ui_som
Low level input/output	ll_som
Memory (real storage) management	mm_som
Ports/semaphores	po_som
Process management/dispatcher/loader	pm_som
Symbol table management	st_som
Table management	tm_som
Transaction management	xm_som
Virtual storage management	vs_som

Organization

All of the MPE XL type definitions will be included in a single SOM file. This SOM file will be created by compiling each subsystem's type declarations separately with the SYMDEBUG compiler option turned on, and using the linkage editor to combine them into a single SOM. PXDB will then be run against that SOM to organize the declarations such that they are usable by the DAT. The major non-MPE XL systems (Database, Data Communications) will also have a single SOM file that is prepared in a similar manner.

Contents

Chapter 2
Configuration Macros

2

Configuration Macros

Introduction

All configuration information is listed in this section.

The macro name is listed next to each item name where a macro is defined.

config_device_ldev

Prints information on all the devices configured on the system by logical device number.

Syntax

```
config_device_ldev
      ([ldev   : INT / U16 = $0] ,
       [detail : INT / U16 = $5] )
```

Parameters

ldev

Allows the user to specify which ldev to print. If 0 or no ldev is passed all ldev's are printed.

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 3 Will print the Ldev #, device type and physical path name.
4 - 5 Will print the Ldev #, device type, Logical Device Manager's Port #, Logical Device Manager's Port Data Area address, Device Manager's Port # and Device Manager's Port

User Discussion

All Ldev numbers are listed in decimal format.

The ports may be examined using the port_info macro. The particular LPT entry may be displayed using the io_lpt macro.

The LDM Port Data area may be formatted using the type 'ldm_server_data_area'. The DM Port data area may be formatted using the type 'ios_port_data_rec' for IO Services section which is first. The Device Manager's section will follow and may be formatted using the Device Manager's type. They are:

```
CS80 disc        - 'ddm_port_data_type'
7974/78 tape     - 'tp_port_data_type'
2680/88 printer  - 'pp_port_data_type'
256X printer     - 'port_data_type'
Console          - 'mux_pda_type'
```

Example

```
$30e ($21) nmdat > config_device_ldev
LDEV#    TYPE                LDM Port   LDM PDA     DM Port    DM PDA
-----    ----                --------   -------     -------    ------
1        IO_DISC             ffffffcc   a.c04d8200  ffffffcd   a.c04b8200
2        IO_DISC             ffffffa9   a.c0608200  ffffffaa   a.c05f8200
3        IO_DISC             ffffffa7   b.80c40200  ffffffa8   b.80c38200
4        IO_DISC             ffffffa5   a.c0638200  ffffffa6   a.c0618200
6        IO_PRINTER          ffffff95   b.80ca0200  ffffff96   b.80c98100
7        IO_TAPE             ffffffc6   a.c04f0200  ffffffc7   b.80528100
8        IO_TAPE             ffffff9b   a.c06e8200  ffffff9c   b.80c80100
9        IO_TAPE             ffffff92   b.80cb0200  ffffff93   a.c07a0100
10       IO_TAPE             ffffff99   a.c06f0200  ffffff9a   b.80c88100
14       IO_DISC             ffffffa3   b.80c58200  ffffffa4   b.80c50200
15       IO_DISC             ffffffa1   a.c06c8200  ffffffa2   a.c06c0200
16       IO_DISC             ffffff9f   b.80c70200  ffffffa0   b.80c68200
17       IO_DISC             ffffff9d   a.c06e0200  ffffff9e   a.c06d8200
19       IO_SERIAL_PRINTER   ffffff97   a.c06f8200  ffffff98   b.80c90100
20       IO_TERMINAL         ffffffcf   b.804b0200  ffffffd0   b.803f8200
100      IO_TERMINAL         ffffff69   b.814d8200  ffffff6a   a.c14d8200
101      IO_TERMINAL         ffffff61   a.c15b8200  ffffff62   b.814f8200

$37f ($0) nmdat > config_device_ldev
LDEV#    TYPE                LDM Port   LDM PDA     DM Pc      DM PDA
-----    ----                --------   -------     -----      ------
1        IO_DISC             ffffffcc   a.c04d8200  ffffft..   a.c04b8200
2        IO_DISC             ffffffa9   a.c0608200  ffffffaa   a.c05f8200
3        IO_DISC             ffffffa7   b.80c40200  ffffffa8   b.80c38200
4        IO_DISC             ffffffa5   a.c0638200  ffffffa6   a.c0618200
6        IO_PRINTER          ffffff95   b.80ca0200  ffffff96   b.80c98100
7        IO_TAPE             ffffffc6   a.c04f0200  ffffffc7   b.80528100
8        IO_TAPE             ffffff9b   a.c06e8200  ffffff9c   b.80c80100
9        IO_TAPE             ffffff92   b.80cb0200  ffffff93   a.c07a0100
10       IO_TAPE             ffffff99   a.c06f0200  ffffff9a   b.80c88100
14       IO_DISC             ffffffa3   b.80c58200  ffffffa4   b.80c50200
15       IO_DISC             ffffffa1   a.c06c8200  ffffffa2   a.c06c0200
16       IO_DISC             ffffff9f   b.80c70200  ffffffa0   b.80c68200
17       IO_DISC             ffffff9d   a.c06e0200  ffffff9e   a.c06d8200
19       IO_SERIAL_PRINTER   ffffff97   a.c06f8200  ffffff98   b.80c90100
20       IO_TERMINAL         ffffffcf   b.804b0200  ffffffd0   b.803f8200
...        ....                ....       .....       ....       .....
...        ....                ....       .....       ....       .....
100      IO_TERMINAL         ffffff69   b.814d8200  ffffff6a   a.c14d8200
222      IO_TERMINAL         fffffcc1   b.83590200  fffffcc2   a.c3510200
223      IO_TERMINAL         fffffcb9   a.c3538200  fffffcba   b.835b0200
224      IO_TERMINAL         fffffcb1   b.835d8200  fffffcb2   a.c3558200
225      IO_TERMINAL         fffffca9   a.c3580200  fffffcaa   b.835f8200
```

```
$30f ($21) nmdat > config_device_ldev(,1)
LDEV#     TYPE                  PATH
-----     ----                  ----
1         IO_DISC               2/4.0.0
2         IO_DISC               2/4.0.1
3         IO_DISC               2/4.0.2
4         IO_DISC               2/4.0.3
6         IO_PRINTER            2/4.2.7
7         IO_TAPE               2/4.2.3
8         IO_TAPE               2/4.2.2
9         IO_TAPE               2/4.3.3
10        IO_TAPE               2/4.2.4
14        IO_DISC               2/4.0.4
15        IO_DISC               2/4.0.5
16        IO_DISC               2/4.0.6
17        IO_DISC               2/4.0.7
19        IO_SERIAL_PRINTER     2/4.2.6
20        IO_TERMINAL           2/4.1.0
100       IO_TERMINAL
101       IO_TERMINAL
102       IO_TERMINAL

$3e7 ($0) nmdat > config_device_ldev(1,2)

LDEV#     TYPE
-----     ----
1         IO_DISC


$3e8 ($0) nmdat > config_device_ldev(,2)

LDEV#     TYPE
-----     ----
1         IO_DISC
2         IO_DISC
3         IO_DISC
4         IO_DISC
6         IO_PRINTER
7         IO_TAPE
8         IO_TAPE
9         IO_TAPE
10        IO_TAPE
14        IO_DISC
15        IO_DISC
16        IO_DISC
17        IO_DISC
19        IO_SERIAL_PRINTER
20        IO_TERMINAL
..          .....
..          .....
100       IO_TERMINAL
224       IO_TERMINAL
225       IO_TERMINAL
```

```
$3e9 ($0) nmdat > config_device_ldev(1,3)

LDEV#    TYPE
-----    ----
1        IO_DISC


$3ea ($0) nmdat > config_device_ldev(1,4)

LDEV#    TYPE                LDM Port  LDM PDA     DM Port   DM PDA
-----    ----                --------  -------     -------   ------
1        IO_DISC             fffffcc   a.c04d8200  fffffcd   a.c04b8200


$3eb ($0) nmdat > config_device_ldev(1,5)

LDEV#    TYPE                LDM Port  LDM PDA     DM Port   DM PDA
-----    ----                --------  -------     -------   ------
1        IO_DISC             fffffcc   a.c04d8200  fffffcd   a.c04b8200


$3ec ($0) nmdat > config_device_ldev(1,1)

LDEV#    TYPE
-----    ----
1        IO_DISC


$3ed ($0) nmdat > config_device_ldev(1,0)

LDEV#    TYPE
-----    ----
1        IO_DISC
```

```
$401 ($0) nmdat > config_device_ldev(,1)

LDEV#     TYPE                 PATH
-----     ----                 ----
1         IO_DISC              2/4.0.0
2         IO_DISC              2/4.0.1
3         IO_DISC              2/4.0.2
4         IO_DISC              2/4.0.3
6         IO_PRINTER           2/4.2.7
7         IO_TAPE              2/4.2.3
8         IO_TAPE              2/4.2.2
9         IO_TAPE              2/4.3.3
10        IO_TAPE              2/4.2.4
14        IO_DISC              2/4.0.4
15        IO_DISC              2/4.0.5
16        IO_DISC              2/4.0.6
17        IO_DISC              2/4.0.7
19        IO_SERIAL_PRINTER    2/4.2.6
20        IO_TERMINAL          2/4.1.0
..          .....
..          .....
224       IO_TERMINAL
225       IO_TERMINAL
```

config_device_path

Prints information on all the devices configured on the system by logical device number.

Syntax

```
config_device_path
        ([path          : STR = ''] ,
        [detail        : INT / U16 = $1] )
```

Parameters

path

This is a string containing the path name to print. If no path is specified, all paths will be printed.

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 2	Will print the Path Name, Manager Name, Port Number of the manager, Port Data Area Pointer, ICS priority and Environment in which the manager will run.
3 - 5	Will execute an 'FV' of the specified path entry. If no path was specified, detail will be automatically set to 1.

User Discussion

The ports may be examined using the port_info macro.

The Manager's Port Data Area may be formatted using the type 'ios_port_data_rec' for IO Services section, which is first. The Manager's section will follow and may be formatted using the Manager's type. They are:

```
CS80 disc          - 'cs80_port_data_type' or 'ddm_port_data_type'
7974/78 tape       - 'tp_port_data_type'
2680/88 printer    - 'pp_port_data_type'
256X printer       - 'port_data_type'
Console            - 'mux_pda_type'
TMUX DAM           - 'mux_pda_type'
HPIB DAM           - 'svr_area_type'
CAM                - 'cio_port_data_type'
```

Example

```
$30d ($21) nmdat > config_device_path
PATH                MANAGER NAME        PORT #    PDA PTR    PRI  ENV
----                ------------        ------    -------    ---  ---
2                   BUS_CONV_MGR        ffffffd2  803e8200     2  MGR_ON_ICS
2/4                 SPECTRUM_CIO_CAM    ffffffd1  c04a8200     6  MGR_ON_ICS
2/4.0               HPIB_DAM            ffffffce  c04b0200     6  MGR_ON_ICS
2/4.0.0             CS80_DISC_DM        ffffffcd  c04b8200     8  MGR_ON_ICS
2/4.0.1             CS80_DISC_DM        ffffffaa  c05f8200     8  MGR_ON_ICS
2/4.0.2             CS80_DISC_DM        ffffffa8  80c38200     8  MGR_ON_ICS
2/4.0.3             CS80_DISC_DM        ffffffa6  c0618200     8  MGR_ON_ICS
2/4.0.4             CS80_DISC_DM        ffffffa4  80c50200     8  MGR_ON_ICS
2/4.0.5             CS80_DISC_DM        ffffffa2  c06c0200     8  MGR_ON_ICS
2/4.0.6             CS80_DISC_DM        ffffffa0  80c68200     8  MGR_ON_ICS
2/4.0.7             CS80_DISC_DM        ffffff9e  c06d8200     8  MGR_ON_ICS
2/4.1               TMUX_DAM_ME         ffffffd0  803f8200     6  MGR_ON_ICS
2/4.1.0             TMUX_DAM_ME         ffffffd0  803f8200     9  MGR_ON_ICS
2/4.2               HPIB_DAM            ffffffc8  80520200     6  MGR_ON_ICS
2/4.2.2             TAPE_7978_DM        ffffff9c  80c80100     a  MGR_ON_STACK
2/4.2.3             TAPE_7978_DM        ffffffc7  80528100     a  MGR_ON_STACK
2/4.2.4             TAPE_7978_DM        ffffff9a  80c88100     a  MGR_ON_STACK
2/4.2.6             PP_DM               ffffff98  80c90100     a  MGR_ON_PFP
2/4.2.7             CIPER_DM            ffffff96  80c98100     a  MGR_ON_PFP
2/4.3               HPIB_DAM            ffffff94  c0798200     6  MGR_ON_ICS
2/4.3.3             TAPE_7978_DM        ffffff93  c07a0100     a  MGR_ON_STACK
2/8                 SPECTRUM_CIO_CAM    ffffff91  c07a8200     6  MGR_ON_ICS
2/8.0               IEEE8023_DAM        ffffff75  81490140     6  MGR_ON_ICS
2/8.0.0             TIO_DTCM            ffffff6c  c14d0000     8  MGR_ON_STACK
2/8.0.1             TIO_FCM             ffffff6b  814b8180     8  MGR_ON_STACK
2/8.0.1.1           TIO_TSPDM           ffffff6a  c14d8200     8  MGR_ON_STACK
2/8.0.2             TIO_FCM             ffffff63  c15b0180     8  MGR_ON_STACK
2/8.0.2.1           TIO_TSPDM           ffffff62  814f8200     8  MGR_ON_STACK
~0                  DCC_SURROGATE       ffffff76  c1458000     f  MGR_ON_STACK
~1                  TIO_NDM             ffffff70  814a8000     f  MGR_ON_STACK

$30e ($21) nmdat > config_device_path('2/4.0')
PATH                MANAGER NAME        PORT #    PDA PTR    PRI  ENV
----                ------------        ------    -------    ---  ---
2/4.0               HPIB_DAM            ffffffce  c04b0200     6  MGR_ON_ICS

$380 ($0) nmdat > config_device_path

PATH                MANAGER NAME        PORT #    PDA PTR    PRI  ENV
----                ------------        ------    -------    ---  ---
2                   BUS_CONV_MGR        ffffffd2  803e8200     2  MGR_ON_ICS
2/4                 SPECTRUM_CIO_CAM    ffffffd1  c04a8200     6  MGR_ON_ICS
2/4.0               HPIB_DAM            ffffffce  c04b0200     6  MGR_ON_ICS
2/4.0.0             CS80_DISC_DM        ffffffcd  c04b8200     8  MGR_ON_ICS
```

```
2/4.0.1          CS80_DISC_DM      ffffffaa  c05f8200   8  MGR_ON_ICS
2/4.0.2          CS80_DISC_DM      ffffffa8  80c38200   8  MGR_ON_ICS
2/4.0.3          CS80_DISC_DM      ffffffa6  c0618200   8  MGR_ON_ICS
2/4.0.4          CS80_DISC_DM      ffffffa4  80c50200   8  MGR_ON_ICS
2/4.0.5          CS80_DISC_DM      ffffffa2  c06c0200   8  MGR_ON_ICS
2/4.0.6          CS80_DISC_DM      ffffffa0  80c68200   8  MGR_ON_ICS
2/4.0.7          CS80_DISC_DM      ffffff9e  c06d8200   8  MGR_ON_ICS
2/4.1            TMUX_DAM_ME       ffffffd0  803f8200   6  MGR_ON_ICS
2/4.1.0          TMUX_DAM_ME       ffffffd0  803f8200   9  MGR_ON_ICS
2/4.2            HPIB_DAM          ffffffc8  80520200   6  MGR_ON_ICS
2/4.2.2          TAPE_7978_DM      ffffff9c  80c80100   a  MGR_ON_STACK
2/4.2.3          TAPE_7978_DM      ffffffc7  80528100   a  MGR_ON_STACK
2/4.2.4          TAPE_7978_DM      ffffff9a  80c88100   a  MGR_ON_STACK
2/4.2.6          PP_DM             ffffff98  80c90100   a  MGR_ON_PFP
2/4.2.7          CIPER_DM          ffffff96  80c98100   a  MGR_ON_PFP
2/4.3            HPIB_DAM          ffffff94  c0798200   6  MGR_ON_ICS
2/4.3.3          TAPE_7978_DM      ffffff93  c07a0100   a  MGR_ON_STACK
2/8              SPECTRUM_CIO_CAM  ffffff91  c07a8200   6  MGR_ON_ICS
2/8.0            IEEE8023_DAM      ffffff75  81490140   6  MGR_ON_ICS
2/8.0.0          TIO_DTCM          ffffff6c  c14d0000   8  MGR_ON_STACK
2/8.0.1          TIO_FCM           ffffff6b  814b8180   8  MGR_ON_STACK
2/8.0.1.1        TIO_TSPDM         ffffff6a  c14d8200   8  MGR_ON_STACK
 ....              ....              ....      ...     .     .....
 ....              ....              ....      ...     .     .....
2/8.0.91.1       TIO_TSPDM         fffffcb2  c35582[illegible]  8  MGR_ON_STACK

2/8.0.92         TIO_FCM           fffffcab  c3578[illegible]   8  MGR_ON_STACK
2/8.0.92.1       TIO_TSPDM         fffffcaa  835f820[illegible] 8  MGR_ON_STACK
~0               DCC_SURROGATE     ffffff76  c1458000   f  MGR_ON_STACK
~1               TIO_NDM           ffffff70  814a8000   f  MGR_ON_STACK
```

console_ldev

Prints the logical device number of the system console.

Syntax

```
console_ldev
```

User Discussion

Prints the logical device number for the system console.

Example

```
$381 ($0) nmdat > console_ldev

SYSTEM CONSOLE AT LDEV #20
```

config_memory

Prints the physical memory configuration information.

Syntax

```
config_memory ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print memory configuration information.

The default is 5. (optional parameter)

User Discussion

For each processor (including the global memory), the following is printed is the summary option is specified:

amount of memory

If the detail option is specified, the following is printed in addition:

detailed memory configuration (controllers/memory cards)
global memory interconnection

Example

```
$382 ($0) nmdat > config_memory

PROCESSOR  GLOBAL MEMORY SIZE
---------  ------------------
              $a000
```

Contents

Chapter 3
Current State Macros

3

Current State Macros

Introduction

The current state includes any data that describes the condition of the machine at the time of the dump. This includes the processor state (running or idle), system failure information, current process information, device information, etc.

file_in_use

Prints all files being used, shared by all process.

Syntax

```
file_in_use([detail:int=5],
            [system_files:bool=false])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 3	Print a list of filenumbers, a filename for each file number, and a list of accessing processes.
4 - 5	Print information plus the access options and file options for each file.

The default is 5. (optional parameter)

system_files

to be supplied later

User Discussion

All entries are printed in file type and filename order.

Example

global_memory

Print information about global memory.

Syntax

```
global_memory([detail:int=1])
```

Parameters"

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print all global memory domains including virtual req_addresses, lengths of objects, and object classes.

The default is 1. (optional parameter)

User Discussion

Information concerning global memory is printed. This will include all virtual req_addresses of global memory objects, the length of each virtual memory object, and the object class of each memory object.

Example

job_current

Prints current jobs by scheduling state and job type.

Syntax

```
job_current ([detail:int=5],
             [scheduling_state:str=''],
             [job_type:str=''])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 2 Print only summary information.

3 - 5 Print detailed information.

4 - 5 Print a list of JOB#, time, user, input done, output device and ci process with header.

The default is 5. (optional parameter)

scheduling_state

A single ASCII character to specify the scheduling state for the jobs to be printed. Examples of this are "s" or "S" for suspended or "w" for waiting. (Other states are terminating and executing.) The default is to print all jobs in all states. (optional parameter).

job_type

A single ASCII character to specify whether to print summary information on batch jobs ("b" or "B"), terminal sessions ("s" or "S"), or both. The default is to print both. (optional parameter).

User Discussion

If the summary option is specified, only the number of jobs in the in each scheduling state and job type is printed. If the detail option is specified the job identification, user name, job input/output devices, job start time, and command interpreter process id is printed for each job in the specified (or all) scheduling state and the specified (or all) job type. These entries are printed in order by job type, scheduling state, and job identification.

Example

```
$36c ($36) nmdat > job_current

  JOB     TIME          USER          INPUT DEVICE  OUTPUT DEVICE   CI PROCESS
 -----  --------  ---------------  ------------  -------------   ----------
#S1     11:06:23  MANAGER .SYS         $14           $14          $36

$36d ($36) nmdat > job_current(2)

BATCH JOBS $0
```

job_oper_req

Print any outstanding operator requests.

Syntax

```
job_oper_req ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 3	Print only summary level information.
4 - 5	Print detailed information.

The default is 5. (optional parameter)

User Discussion

Each request is printed in order by time requested. The time, process id, and message is printed with each request if the detail option is specified. If the summary option is specified, only the total number outstanding operator reply requests is printed.

Example

machine_state

Prints the state of the processor and general dump information.

Syntax

```
machine_state ([detail:int=5])
```

Parameter

detail

0	Suppress printing.
1 - 5	Print out MPE XL version, CPU console ldev and current registers information.

The default is 5. (optional parameter)

User Discussion

Indicates whether the machine was running or idle, if the machine was st[illegible] because of a system failure, the system failure identification is printed. The time that the m[illegible] was dumped is printed.

Example

```
$2e8 ($36) nmdat > machine_state

MPE/XL VERSION:  9.5..6C          CPU: PROCESS_RUNNING

SYSTEM CONSOLE AT LDEV #20

CURRENT REGISTERS:

R0 =00000000 c01066e0 00293a9c 400114a0 R4 =c2c20000 00101000 40011484 0000011c
R8 =40011432 00000010 00000013 fffffffe0 R12=0000011c 0000011c 00000010 00000010
R16=0000fff0 40201f9c 00000003 4020368c R20=0000000a 00293a94 c01061a0 0000011c
R24=00000079 000c006d fc190064 c0200008 R28=000c006d 000cf000 40202190 00000000

IPSW=0004f80b=jthlnxbCvmrQpDI  PRIV=0   SAR=0018 PCQF=a.2975bc   a.2975c0

SR0=0000000a 0000000a 00000000 00000000 SR4=0000000a 0000011c 0000000b 0000000a
TR0=005e4a00 00624a00 00096818 00000000 TR4=400114a0 c2c20000 00101000 40011484
PID1=010a=0085(W) PID2=0064=0032(W)      PID3=0000=0000(W) PID4=0000=0000(W)

RCTR=00000000 ISR=0000000a IOR=00000000 IIR=00000000 IVA=0008c000 ITMR=35c8102d
EIEM=ffffffff EIRR=00000000 CCR=0000
```

process_dispatcher

Prints relevant dispatcher information.

Syntax

```
process_dispatcher ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1	Print only dispatcher scheduling information.
2	Print dispatcher scheduling information and certain ICS/dispatcher relevant information.
3	Print all information plus the dispatch queue.
4 - 5	Print all information plus the dispatcher work queue.

The default is 5. (optional parameter)

User Discussion

If the summary option is specified, the processor state and the dispatcher queue will be printed. If the detailed option is specified, the processor state, dispatcher queue, scheduling information, queue quantum, queue base/limit information, and miscellaneous dispatcher information will be printed.

Macros used

```
hpepm_disp_global
hpepm_ics_header
hpepm_disp_queue
```

Example

```
$30d ($36) nmdat > process_dispatcher

                        PROCESSOR STATE
                        ---------------
                        PROCESS_ACTIVE

                    === SCHEDULING INFORMATION ===

 READY    ACTIVE    ACTIVE    PENDING     PENDING
 COUNT     PIN       PRI       PIN         PRI
 -----    ------    ------    -------     -------
$0        $36       $bb2       $7ffd      $0

     ES             DS             CS             CS             CS
  QUANTUM        QUANTUM        QUANTUM       QUANTUM MAX    QUANTUM MIN
 ---------      ---------      -------       -----------    -----------
$1a39de0       $1a39de0       $1a39de0      $1a39de0       $68e778

 ES PRIORITY       DS PRIORITY       CS PRIORITY
 BASE    LIM       BASE    LIM       BASE    LIM
 ----    ---       ----    ---       ----    ---
$3e8     $1        $7d0    $3e9      $bb8    $7d1

                    === MISCELLANEOUS INFORMATION ===

DISP DISABLE    DISP DISABLE     ICS DISP     ICS NESTED     INT DISABLE
   COUNT            PIN         DSABL CNT      INT CNT         COUNT
------------    ------------    ---------    ----------    -----------
$0                 $36             $0             $0          $0

DISPATCH QUEUE INFO:       PIN      PRIORITY
                           ---      --------
              HEAD ===>    $36        $bb2
```

process_wait

Prints all processes waiting on something (semaphore, I/O, etc.)

Syntax

```
process_wait ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 3	Print the wait type and a list of pins waiting on that type of event.
4 - 5	Print information by pin.

The default is 5. (optional parameter)

User Discussion

If detail is 0 than nothing is printed. If detail is 1 then a list of wait types with a list of pins who are waiting on each wait type is printed.

Example

```
$2e8 ($36) nmdat > process_wait

PROCESS                 WAIT TYPE
-------                 ---------
$1                      IPC
$2                      IPC
$3                      IPC
 ..                     ...
 ..                     ...
$33                     IPC
$34                     IPC
$35                     IPC
```

res_all

Prints all high-level information about resources.

Syntax

```
res_all([detail:int=5])
```

Parameter

detail

0	Suppress printing.
1 - 5	Print all high level information about resources. Print all of the critical resources held by a process by resource. Print all critical resources where there is a process waiting in the resource. Print all deadlocked resources by type.

The default is 5. (optional parameter)

User Discussion

Calls `res_deadlock`, `res_crit_held`, and `res_crit_waiting`.

Macros Used

```
res_crit_held;
res_crit_waiting;
```

Example

```
$30f ($36) nmdat > res_all

PINs owning semaphores:
No PINs own semaphores.

PINs owning SIRs:
No PINs own SIRs.

PINs waiting on Semaphores:
No PINs are waiting on any semaphores.

PINs waiting on SIRs:
No PINs waiting on any SIRs.

Getting semaphore info for ALL pins ...
Getting SIR info for ALL pins ...

 No Deadlock detected.
```

res_crit_held

Prints all of the critical resources held by a process by resource type.

Syntax

```
res_crit_held ([detail:int=5])
```

Parameters

detail

0	Suppress printing.
1 - 5	Prints all of the critical resources held by a process.

The default is 5. (optional parameter)

User Discussion

The resource type (SIR, critical semaphore, port, etc.) is printed with the process id of the process holding the resource.

Example

```
$310 ($36) nmdat > res_crit_held

PINs owning semaphores:
No PINs own semaphores.

PINs owning SIRs:
No PINs own SIRs.
```

res_crit_waiting

Prints all critical resources where there is a process waiting on the resource.

Syntax

```
res_crit_waiting ([detail:int=5])
```

Parameters

detail

0	Suppress printing.
1 - 5	Prints all critical resources where there is a process waiting on the resource (optional parameter).

The default is 5. (optional parameter)

User Discussion

The entries are printed by resource type (see res_crit_held). The resource type and a list of process ids (in the order they are waiting) are printed for each critical resource type.

Example

```
$311 ($36) nmdat > res_crit_waiting
PINs waiting on Semaphores:
No PINs are waiting on any semaphores.

PINs waiting on SIRs:
No PINs waiting on any SIRs.
```

Contents

Chapter 4
Recent Events Macros

4

Recent Events Macros

Introduction

Recent events are all data about any normal or unusual conditions which have happened recently. This does not include the 'current state data' - which is documented in another section.

Most of this will be obtained from footprint buffers, other trace buffers, and error logs.

event_ci_history

Prints all CI redo stacks for all live processes.

Syntax

```
event_ci_history ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1	Print only the last five CI commands executed for all processes.
2 - 5	Print the entire CI redo stack for all processes.

The default is 5. (optional parameter)

User Discussion

Prints CI redo stacks. Either print the last five commands that all processes have executed or print the whole CI redo stack for all processes.

Example

```
$2e2 ($36) nmdat >EVENT_CI_HISTORY

SESSION NUMBER: #S1
SESSION NAME:    ,MANAGER.SYS,PUB

CI HISTORY STACK FOR PIN: $36
redo_stk_addr :
46516818

redo_stk[ $1 ] =  ''
redo_stk[ $2 ] =  ''
   .....           ..
   .....           ..
redo_stk[ $13 ] =  ''
redo_stk[ $14 ] =  ''
```

```
$2e4 ($36) nmdat > event_ci_history(1)

SESSION NUMBER: #S1
SESSION NAME:    ,MANAGER.SYS,PUB

CI HISTORY STACK FOR PIN: $36

redo_stk_addr :
46516818

redo_stk[ $1 ] =  ''
redo_stk[ $2 ] =  ''
redo_stk[ $3 ] =  ''
redo_stk[ $4 ] =  ''
redo_stk[ $5 ] =  ''
```

event_footprint

Prints the global footprint information.

Syntax

```
event_footprint ([detail:int+=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 5 Print the formatted entries in the global footprint table.

The default is 5. (optional parameter)

User Discussion

Prints the global HPE footprint table. Formats each entry.

Example

```
$2e3 ($36) nmdat > event_footprint

PIN   TIMESTAMP   SUBSYS  TRACE1  TRACE2  TRACE3   TRACE4   TRACE5  TRACE6
---   ---------   ------  ------  ------  ------   ------   ------  ------
$e2   $fa         $0      $6a     $0      $0       $0       $0      $0
$7fff $348a4671   $0      $c007b  $36     $b8300c0 $c7400380$0     $0
$36   $348a9196   $0      $c007b  $36     $9000001 $22      $0      $0
$7fff $348a992a   $0      $3007b  $36     $7ffd    $0       $0      $0
 ...    ....      .        ...     ..      ...      .        .       .
 ...    ....      .        ...     ..      ...      .        .       .
$7fff $3480edb3   $0      $3007b  $36     $7ffd    $0       $0      $0
$7fff $34816c11   $0      $6007b  $36     $bb4     $0       $0      $0
$7fff $34816de9   $0      $4007b  $0      $0       $0       $0      $0
```

event_io_trace

Prints the I/O monitor table information.

Syntax

```
event_io_trace ([detail:int=5])
```

Parameters

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 3	Print all entries in the I/O monitor table.
	This includes the time, the I/O action, and the LLIO transaction number.
4	Print info plus the sender's port number, and the type specific information.
5	Print all information plus the io server history of the most recent I/O errors.

The default is 5. (optional parameter)

User Discussion

Prints entries in the I/O monitor table. If detail level one is specified, the n only the last five entries in the table are printed. All other detail levels print the whole monitor table.

Example

```
$2e5 ($36) nmdat > event_io_trace(1)

TIME  TRANSACTION    ACTION TYPE
----  -----------    -----------
$0    $0             $0
$0    $0             $0
$0    $0             $0
$0    $0             $0
$0    $0             $0
```

event_process

Prints the process local footprint table.

Syntax

```
event_process ([detail:int=5],[pin_num:int=0]),
```

Parameters

pin

If pin is specified than the footprint table entries for that pin only will be displayed.

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 5 Print the formatted entries from the process local footprint table.

The default is 5. (optional parameter)

User Discussion

Prints the process local footprint table. If a pin is specified then only those entries that correspond to that pin will be displayed. If no pin is specified than the whole table will be printed.

Example

```
$2e9 ($36) nmdat > event_process
No footprint pointer for PIN $1
No footprint pointer for PIN $2
No footprint pointer for PIN $3
 ..............                ..
 ..............                ..
No footprint pointer for PIN $35
No footprint pointer for PIN $36
```

event_process_errors

Prints process error stacks.

Syntax

```
event_process_errors ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Print out header only.
1 - 5	Print only subsystem number, subsystem error #, and subsystem name.

The default is 5. (optional parameter)

User Discussion

Example

```
$2ea ($36) nmdat > event_process_errors

ERROR STACK FOR PIN:$1

NO ERRORS IN STACK

 .......
 .......

ERROR STACK FOR PIN:$a

NO ERRORS IN STACK

ERROR STACK FOR PIN:$b

ERROR STACK (Entries displayed in most recent to least recent order):
```

```
SUBSYSTEM      SUBSYSTEM          SUBSYSTEM
NUMBER         ERROR NUMBER       NAME
---------      ------------       ---------
$7a            $b4                Port (IPC) Facility
$7a            $352               Port (IPC) Facility
ERROR STACK FOR PIN:$c

NO ERRORS IN STACK


ERROR STACK FOR PIN:$d

ERROR STACK (Entries displayed in most recent to least recent order):

SUBSYSTEM      SUBSYSTEM          SUBSYSTEM
NUMBER         ERROR NUMBER       NAME
---------      ------------       ---------
$7a            $b4                Port (IPC) Facility
$7a            $b4                Port (IPC) Facility
$7a            $b4                Port (IPC) Facility
$7a            $b4                Port (IPC) Facility

ERROR STACK FOR PIN:$e
ERROR STACK (Entries displayed in most recent to least recent order):

SUBSYSTEM      SUBSYSTEM          SUBSYSTEM
NUMBER         ERROR NUMBER       NAME
---------      ------------       ---------
$7a            $b4                Port (IPC) Facility
ERROR STACK FOR PIN:$f

ERROR STACK (Entries displayed in most recent to least recent order):

SUBSYSTEM      SUBSYSTEM          SUBSYSTEM
NUMBER         ERROR NUMBER       NAME
---------      ------------       ---------
$7a            $b4                Port (IPC) Facility

ERROR STACK FOR PIN:$10

NO ERRORS IN STACK

 ............
 ............

ERROR STACK FOR PIN:$18

ERROR STACK (Entries displayed in most recent to least recent order):
```

```
SUBSYSTEM      SUBSYSTEM         SUBSYSTEM
NUMBER         ERROR NUMBER      NAME
---------      ------------      ---------
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility
$7a            $b4               Port (IPC) Facility

ERROR STACK FOR PIN:$19

NO ERRORS IN STACK


  ..............
  ..............

ERROR STACK FOR PIN:$36
ERROR STACK (Entries displayed in most recent to least recent order):

SUBSYSTEM      SUBSYSTEM         SUBSYSTEM
NUMBER         ERROR NUMBER      NAME
---------      ------------      ---------
$6d            $c                Trap Handler
```

Contents

Chapter 5
File System Macros

5

File System Macros

Introduction

The file system will be able to get file information through the specification of one of the following:

```
process id, file number
filename
Unique File ID (UFID)
```

Only open files will be able to be accessed using these macros. Recently closed files may also be available for access if the control blocks representing those files are still available. This cannot be predicted for any given file.

Accessing files by filename or UFID will take a long time, since a large system table (VSOD/GUFD) will need to be searched sequentially for each access.

Syntactic Conventions

The UFID will be displayed and entered by the user in the following format:

```
volumesetname:volumename:index
```

Note the index which is displayed is whatever radix is current. An example UFID might be:

```
HPE_SYSTEM_VOLUME_SET:MEMBER2:$2db
```

File names will always be a single string specified with each component of the filename separated by a period (as in MPE).

All file types share some common data structures in the current MPE XL File System, but there are other structures unique to certain file types. In particular, file types which are handled in Compatibility Mode use a set of CM data structures which are a carryover from MPE V. CM file types also use some of the NM structures (especially disc files). A separate set of macros has been defined to deal with the CM File system.

Definitions of native mode file data structures:
GDPD - Global Data Pointer Descriptor (multi-access NM disc files only)
GUFD - Global Unique File Descriptor (CM and NM disc files only)
PLFD - Process Logical File Descriptor (all files, CM and NM)
UFID - Unique File IDentifier (disc files only)
FLAB - File LABel (resident on disc for all disc files)

Definitions of compatibility mode file system data structures:

AFT - Active File Table (all CM files)
LACB - Logical Access Control Block (multiacc CM files only)

PACB - Physical Access Control Block (all CM files)
FCB - File Control Block (CM disc files only)
FMAVT - File Multi-Access Vector Table (multiacc CM disc files only)

fs_acc_attr

Print the file access attributes for specified file. Return the aoptions in effect for the PLFD.

Syntax

```
aoptions:u16 = fs_acc_attr (plfd_ptr:ptr,
                            [detail:int=0])
```

Parameters

aoptions
The access options are returned here.

plfd_ptr
A pointer to the PLFD entry for the file (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter). The default is 0.

0	Suppresses printing.
1 - 2	Print only the access options.
3	Print more access options.
4	Print full access attributes.
5	Print full access attributes with the pin/fnum on each line.

User Discussion

Returns and prints the access attributes associated with the specified file.

Macros Used

```
fs_aoptions
fs_is_cm_file
fs_format_aoptions
fs_access1
fs_access2
fs_gufd
fs_access3
fs_pacb
fs_fcb
```

Example

```
$nmdat> fs_acc_attr(plfdptr,1)

Access options:          OUTPUT,NOMR,NOLOCK,EXC,BUF,NOMULTI,WAIT,NOCOPY
$nmdat> fs_acc_attr(plfdptr2,5)

Native Mode file

Access options:          EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
Access method:           $0
Last error number:       $0
File pointer:            $0
Num of multiacc users:   $4
Eof pointer:             $fbf00

Total num of readers:    $4
Total num of writers:    $4
```

fs_acc_tree

Print all of the jobs and processes accessing the specified file in a tree fashion with certain access information. Return the list of pins.

Syntax

```
pin_list:str = fs_acc_tree (gufd_ptr:ptr,
                            [detail:int=0],
                            [flab_resident:bool=true])
```

Parameters

pin_list
A list of pins sharing access to the file separated by blanks such that they can be processed with the FOREACH DAT function.

gufd_ptr
A pointer to the GUFD entry for the file (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppress printing, just return pin_list.
1	Print filename, job/session and pins.
2	Print filename, j/s and pins with access options from GDPD.
3 - 4	Print filename, j/s and pins with access options from GDPD and PLFD.
5	Print filename and permanent characteristics along with the access attributes for all accessors.

flab_resident
A flag indicating whether the File Label entry for the file is in memory. If false, no attempt will be made to access the FLAB. (optional parameter)

User Discussion

Searches the VSOD/GUFD table for a match. Only files shared through Native mode structures will work (goes thru GDPD entries).

Macros Used

```
fs_next_gdpd
fs_next_plfd
fs_perm
fs_2_acc_tree
fs_4_acc_tree
fs_aoptions
fs_format_aoptions
```

Example

```
$nmdat> wl fs_acc_tree(gufdptr)

$2a $27 $24 $23
$nmdat> fs_acc_tree(gufdptr,5)


Device type:              $0
Record type:              $0
File type:                $0

File options:             SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
File code:                $408
Record size:              $100
Block size:               $100
Record limit:             $fbf00

Share mode:               SHARED GMULTI
Num of users this mode:   $4
File pointer:             $0
Job:                      $a

    Pin $2a  (#J8)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0

    Pin $27  (#J10)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0
    Pin $24  (#J9)

    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0

    Pin $23  (#J10)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0
```

fs_access_attributes

Print the file access attributes for specified file. Return the aoptions in effect for this accessor.

Syntax

```
aoptions:u16 = fs_access_attributes ([pin_num:int=0],
                                      fnum:int,
                                     [detail:int=5],
                                     [error_parm:str='pad'])
```

Parameters

aoptions
The access options are returned here.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 2	Print only the access options.
3	Print more access options.
4	Print full access attributes.
5	Print full access attributes with the pin/fnum on each line.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Returns and prints the access attributes associated with the specified file (calls fs_acc_attr).

Macros Used

```
fs_plfd
fs_acc_attr
```

Example

```
$nmdat> fs_access_attributes(21,8)

pin $21 file number $8

Native Mode file


Access options:          INPUT,MR,LOCK,SHR,NOBUF,NOMULTI,WAIT,NOCOPY
Access method:           $2
Last error number:       $0
File pointer:            $0
Num of multiacc users:   $1
Eof pointer:             $e94900
Total num of readers:    $19
Total num of writers:    $1

$nmdat> fs_access_attributes(1d,9,1)

Access options:          OUTPUT,NOMR,NOLOCK,EXC,BUF,NOMULTI,WAIT,NOCOPY
```

fs_access_tree_file

Print all of the jobs and processes accessing the specified file in a tree fashion with certain access information. This macro may be very slow as the entire VSOD may need to be scanned.

Syntax

```
pin_list:str = fs_access_tree_file (filename:str,
                                    [detail:int=5],
                                    [error_parm:str='pad'])
```

pin_list
A list of pins sharing access to the file separated by blanks such that they can be processed with the FOREACH DAT function.

filename
the filename of the file to be displayed. See the standard filename formatting convention above for a description of how this parameter is to be formatted (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppress printing, just return pin_list.
1	Print filename, job/session and pins.
2	Print filename, j/s and pins with access options from GDPD.
3 - 4	Print filename, j/s and pins with access options from GDPD. and PLFD.
5	Print filename and permanent characteristics along with the access attributes for all accessors.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Searches the VSOD/GUFD table for a match. Will only work when the file is a disc file whose file label is in memory at the time of the dump. (Calls fs_fname_to_gufd ans fs_acc_tree).

Macros Used

```
fs_fname_to_gufd
fs_acc_tree
```

Example

```
$nmdat> fs_access_tree_file('xc1.lib.sys')

Device type:              $0
Record type:              $0
File type:                $0
File options:             SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
File code:                $408
Record size:              $100
Block size:               $100
Record limit:             $fbf00

Share mode:               SHARED GMULTI
Num of users this mode:   $4
File pointer:             $0
Job:                      $a

    Pin $2a  (#J8)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0

    Pin $27  (#J10)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0

    Pin $24  (#J9)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0

    Pin $23  (#J10)
    Access options:       EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:        $0
    Last error number:    $0
```

fs_addr

Print the virtual address space values of the file. Return a pointer to the virtual address of the file.

Syntax

```
address:ptr = fs_addr (gufd_ptr:ptr,
                       [detail:int=1]
```

Parameters

address
The beginning virtual address of the file is returned here.

gufd_ptr
A pointer to the GUFD entry for the file (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Enables printing.

User Discussion

Return and print the virtual address associated with the specified file. Also print the ending virtual byte address.

Macros Used

fs_vsod

Example

```
$nmdat> fs_addr(gufdptr,5)

Virtual address:          $35.0
Ending virt addr:         $35.fbeff
```

fs_address

Print the virtual address space values of the file. Return a pointer to the virtual address of the file.

Syntax

```
address:ptr = fs_address ([pin_num:int=0],
                          fnum:int,
                         [detail:int=5],
                         [error_parm:str='pad'])
```

Parameters

address
The beginning virtual address of the file is returned here.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Prints detailed information.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Return and print the virtual address associated with the specified file. Also print the ending virtual byte address. (Calls fs_addr).

Macros Used

```
fs_plfd
fs_gufd
fs_addr
```

Example

```
$nmdat> fs_address(21,8)

Virtual address:          $20.0
Ending virt addr:         $20.e948ff
```

fs_aft

Return a pointer to the CM AFT entry corresponding to the specified PLFD entry for a CM file.

Syntax

```
aft_ptr:ptr = fs_aft
              (plfd_ptr:ptr)
```

Parameters

aft_ptr
The virtual address of the CM AFT entry which corresponds to the specified PLFD entry is returned.

plfd_ptr
The virtual address of the PLFD (required).

Macros Used

- objcl_number
- stderr
- fs_cmstack
- fs_aftcm

Example

```
$nmdat> wl fs_aft(plfdptr)
$fd.40011f8c
```

fs_aftcm

Return a pointer to the CM AFT entry corresponding to the specified CM stack and MPE V file number.

Syntax

```
aft_ptr:ptr = fs_aftcm (stack_ptr:ptr, file_num:int=1)
```

Parameters

aft_ptr
The virtual address of the CM AFT entry which corresponds to the specified file number is returned. If the filenum does not correspond to a valid open file, zero will be returned.

stack_ptr
The virtual address of the CM stack in which the AFT for the file resides (required).

file_num
The CM file number of the file. Default file num is 1.

User Discussion

If no file num is specified, the first file opened will be returned. If the file num does not correspond to a valid open file, zero will be returned.

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_aftcm(cmstkptr,2)
$fd.40011fe0
```

fs_all_files

Print information about all files associated with a given process. If the system_files parm is false, only file numbers greater than 7 will be displayed.

Syntax

```
fs_all_files ([pin_num:int=pin],
              [detail:int=5],
              [system_files:bool=false],
              [error_parm:str='pad'])
```

Parameters

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter). This parameter is passed to the called macros (which accept a detail parameter).

system_files
If this flag is false, only the non-system files open by the process will be displayed. That is, only file numbers greater than 7. Otherwise all files starting with file number 0 will be displayed.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. Note that the macros are set up in such a way as to not terminated on errors caused by the File Label Table not being resident in memory at the time of the dump. These errors will be printed, and execution will continue regardless of the error_parm passed in. (optional parameter)

User Discussion

Calls fs_file for every file open by that process.

Macros Used

```
fs_file
fs_next_fnum
```

Example

```
$nmdat> fs_all_files(1b,1,true)

Pin $1b, fnum $0

$STDIN

Access options:          INPUT,NOMR,NOLOCK,SHR,NOBUF,MULTI,WAIT,NOCOPY

File options:            SYS,ASCII,$STDIN,U,NOCCTL,DEQ,STD,NOLABEL

Pin $1b, fnum $1

$STDLIST

Access options:          OUTPUT,NOMR,NOLOCK,SHR,NOBUF,MULTI,WAIT,NOCOPY

File options:            NEW,ASCII,$STDLIST,U,CCTL,DEQ,STD,NOLABEL

Pin $1b, fnum $2

$NULL

Access options:          INPUT,NOMR,NOLOCK,DEF,BUF,NOMULTI,WAIT,NOCOPY

File options:            NEW,BINARY,$NULL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $3

$NULL

Access options:          INPUT,NOMR,NOLOCK,DEF,BUF,NOMULTI,WAIT,NOCOPY

File options:            NEW,BINARY,$NULL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $4

Access options:          INPUT,NOMR,NOLOCK,SHR,BUF,NOMULTI,WAIT,NOCOPY

File options:            SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $5
```

```
Access options:         INPUT,NOMR,NOLOCK,SHR,BUF,NOMULTI,WAIT,NOCOPY
File options:           SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $6

Access options:         INPUT,NOMR,NOLOCK,SHR,BUF,NOMULTI,WAIT,NOCOPY
File options:           SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $7
Access options:         INPUT,NOMR,NOLOCK,SHR,BUF,NOMULTI,WAIT,NOCOPY
File options:           NEW,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $8
JSMAIN.PUB.SYS
Access options:         INPUT,MR,LOCK,SHR,NOBUF,NOMULTI,WAIT,NOCOPY
File options:           SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $9
SL.PUB.SYS
Access options:         INPUT,MR,LOCK,SHR,NOBUF,NOMULTI,WAIT,NOCOPY
File options:           SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $a

Access options:         IN/OUT,NOMR,NOLOCK,DEF,NOBUF,NOMULTI,WAIT,NOCOPY
File options:           NEW,ASCII,FORMAL,U,CCTL,FEQ,STD,NOLABEL

Pin $1b, fnum $b
```

```
Access options:          INPUT,NOMR,NOLOCK,DEF,NOBUF,NOMULTI,WAIT,NOCOPY

File options:            NEW,ASCII,FORMAL,U,NOCCTL,FEQ,STD,NOLABEL


$nmdat> fs_all_files(14)

Pin $14, fnum $8

JSMAIN.PUB.SYS

Native Mode file

Access options:          INPUT,MR,LOCK,SHR,NOBUF,NOMULTI,WAIT,NOCOPY
Access method:           $2
Last error number:       $0
File pointer:            $0
Num of multiacc users:   $1
Eof pointer:             $8800
Total num of readers:    $8
Total num of writers:    $0

Device type:             $0
Record type:             $0
File type:               $0

File options:            SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL
File code:               $405
Record size:             $100
Block size:              $100
Record limit:            $8800

ldev:                    $1
ufid:                    HPE_SYSTEM_VOLUME_SET:MEMBER1:$8a

Virtual address:         $89.0
Ending virt addr:        $89.87ff

ext blk    extent                  vs          # sectors/      virt sect
   #          #         sector     index         extent       num in file
-------    ------      ------     -----       ----------     -----------
 $1         $1         $30000      $1            $90            $0
```

```
Pin $14, fnum $9

SL.PUB.SYS

Native Mode file

Access options:          INPUT,MR,LOCK,SHR,NOBUF,NOMULTI,WAIT,NOCOPY

Access method:           $2
Last error number:       $0
File pointer:            $0
Num of multiacc users:   $1
Eof pointer:             $e94900
Total num of readers:    $19
Total num of writers:    $1

Device type:             $0
Record type:             $0
File type:               $0

File options:            SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL
File code:               $407
Record size:             $100
Block size:              $100
Record limit:            $e94900

ldev:                    $1
ufid:                    HPE_SYSTEM_VOLUME_SET:MEMBER1:$81

Virtual address:         $20.0
Ending virt addr:        $20.e948ff

ext blk   extent                    vs        # sectors/     virt sect
   #        #        sector        index        extent      num in file
-------   ------     ------        -----      ----------    -----------
 $1        $1       $20990         $1           $800          $0
 $1        $2       $21190         $1           $800          $800
 $1        $3       $21990         $1           $800          $1000
 $1        $4       $22190         $1           $800          $1800
 $1        $5       $22990         $1           $800          $2000
 $1        $6       $23190         $1           $800          $2800
 $1        $7       $23990         $1           $800          $3000
 $1        $8       $24190         $1           $800          $3800
 $1        $9       $24990         $1           $800          $4000
 $1        $a       $25190         $1           $800          $4800
 $1        $b       $25990         $1           $800          $5000
 $1        $c       $26190         $1           $800          $5800
 $1        $d       $26990         $1           $800          $6000
 $1        $e       $27190         $1           $800          $6800
 $1        $f       $27990         $1           $800          $7000
 $1        $10      $28190         $1           $800          $7800
```

$1	$11	$28990	$1	$800	$8000
$1	$12	$29190	$1	$800	$8800
$1	$13	$29990	$1	$800	$9000
$1	$14	$2a190	$1	$800	$9800
$2	$1	$2a990	$1	$800	$a000
$2	$2	$2b190	$1	$800	$a800
$2	$3	$2b990	$1	$800	$b000
$2	$4	$2c190	$1	$800	$b800
$2	$5	$2c990	$1	$800	$c000
$2	$6	$2d190	$1	$800	$c800
$2	$7	$2d990	$1	$800	$d000
$2	$8	$6b6e0	$1	$800	$d800
$2	$9	$6bee0	$1	$800	$e000
$2	$a	$69220	$1	$150	$e800

fs_aoptions

Return the access options for the currently open file referred to by the specified PLFD.

Syntax

```
aoptions:u16 = fs_aoptions (plfd_ptr:ptr,
                            [flab_resident:bool=true]))
```

Parameters

aoptions
The aoptions word as specified in the PLFD.

plfd_ptr
A pointer to the PLFD entry for the file (required parameter).

User Discussion

The aoptions are contained in a word in the PLFD.

Macros Used

none

Example

```
$nmdat> wl fs_aoptions(plfdptr):'%'
%101
```

fs_cmcb

Return the virtual address of the CM control block given the CM data segment/CB offset from the AFT.

Syntax

```
cb_ptr:ptr = fs_get_cmcb (cbtdst:int,
                          cbtoff:int)
```

Parameters

cb_ptr
The virtual address of the control block (PACB, LACB, or FCB) is returned.

cbtdst
The CM data segment number containing the control block, as given in the AFT entry (required).

cbtoff
The CM word offset into the control block table for the entry as given in the AFT (required).

Macros Used

```
objcl_number
stderr
fs_cmcbt
```

Example

```
$nmdat> wl fs_cmcb(%160,%10)
$b.83360020
```

fs_cmcbt

Return the virtual address of the CM file Control Block Table given a pointer to the data segment in which the CBT resides.

Syntax

```
cmcbt_ptr:ptr = fs_get_cmcbt(dst_ptr:ptr)
```

Parameters

cmcbt_ptr
The virtual address of the head of the CBT.

dst_ptr
The virtual address of the start of the CM data segment containing the CBT (required).

Macros Used

```
objcl_number
 stderr
 fs_pxfile
```

Example

```
$nmdat> wl fs_cmcbt(cmstkptr)
$fd.40011d8c
```

fs_cmstack

Return a pointer to the CM stack which contains the AFT for the file corresponding to the given PLFD.

Syntax

```
stack_ptr:ptr = fs_cmstack (plfd_ptr:ptr)
```

Parameters

stack_ptr
virtual address pointer to the CM stack where the file is opened.

plfd_ptr
The virtual address of the PLFD (required).

Macros Used

```
objcl_number
stderr
```

Example s:

```
$nmdat> wl fs_cmstack(plfdptr)
$fd.40011cb0

$nmdat> var cmstkptr = fs_cmstack(plfdptr)
```

fs_dev

Print the device and volume (if applicable) information for the file. Return the logical device number.

Syntax

```
ldev:int = fs_dev (plfd_ptr:ptr,
                   [detail:int=0])
```

Parameters

ldev
An integer specifying the logical device identification for the device the file resides on.

plfd_ptr
A pointer to the PLFD entry for the file (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Enables printing.

User Discussion

Calls fs_ldev.

Macros Used

```
fs_ldev
fs_gufd
delete_blanks
```

Example

```
$nmdat> fs_dev(plfdptr,5)

ldev:                    $7

$nmdat> wl fs_dev(plfdptr2,1)
$1
```

fs_device

Print the device and volume (if applicable) information for the file. Return the logical device number.

Syntax

```
ldev:int = fs_device ([pin_num:int=0],
                      num:int,
                      [detail:int=5],
                      [error_parm:str='pad'])
```

Parameters

ldev
An integer specifying the logical device identification for the device the file reside on.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Enables printing.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Return and print the virtual address associated with the specified file. Also print the ending virtual byte address. (Calls fs_dev).

Macros Used

```
fs_plfd
fs_plfd
fs_dev
```

Example

```
$nmdat> fs_device(1d,9)

ldev:                   $7

$nmdat> fs_device(21,8)

ldev:                   $1
ufid:                   HPE_SYSTEM_VOLUME_SET:MEMBER1:$81
```

fs_disc_alc

Print the disc allocation map of the file (if applicable). Return the File Label Table entry pointer for the file.

Syntax

```
flab_ent_ptr:ptr = fs_disc_alc  (gufd_ptr:ptr,
                                 [detail:int=0],
                                 [error_parm:str='pa'])
```

Parameters

flab_ent_ptr
A pointer to the File Label Table entry for the file is returned.

gufd_ptr
A pointer to the GUFD entry for the file (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppress printing.
1 - 5	Prints disc allocation information.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message and add the message to the error stack. (optional parameter)

User Discussion

Goes through File Label Table examining the file's extent blocks.

Macros Used

```
fs_flab
stderr
fs_2_disc_alc
objcl_number
```

Example

```
$nmdat> fs_disc_alc(gufdptr,5)

ext blk    extent                    vs          # sectors/     virt sect
   #         #          sector       index         extent      num in file
-------    ------     ------       -----       ----------     -----------
 $1         $1         $b1de0       $2            $fc0          $0
```

fs_disc_alloc

Print the disc allocation map of the file (if applicable). Return the File Label Table entry pointer for the file.

Syntax

```
flab_ent_ptr = fs_disc_alloc ([pin_num:int=0],
                               fnum:int,
                              [detail:int=5],
                              [error_parm:str='pad'])
```

Parameters

flab_ent_ptr
A pointer to the File Label Table entry for the file is returned.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppress printing.
1 - 5	Prints disc allocation information.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Calls `fs_disc_alc`.

Macros Used

`fs_plfd`
`fs_gufd`
`fs_disc_alc`

Example

```
$nmdat> fs_disc_alloc(21,8)
```

ext blk #	extent #	sector	vs index	# sectors/ extent	virt sect num in file
$1	$1	$20990	$1	$800	$0
$1	$2	$21190	$1	$800	$800
$1	$3	$21990	$1	$800	$1000
$1	$4	$22190	$1	$800	$1800
$1	$5	$22990	$1	$800	$2000
$1	$6	$23190	$1	$800	$2800
$1	$7	$23990	$1	$800	$3000
$1	$8	$24190	$1	$800	$3800
$1	$9	$24990	$1	$800	$4000
$1	$a	$25190	$1	$800	$4800
$1	$b	$25990	$1	$800	$5000
$1	$c	$26190	$1	$800	$5800
$1	$d	$26990	$1	$800	$6000
$1	$e	$27190	$1	$800	$6800
$1	$f	$27990	$1	$800	$7000
$1	$10	$28190	$1	$800	$7800
$1	$11	$28990	$1	$800	$8000
$1	$12	$29190	$1	$800	$8800
$1	$13	$29990	$1	$800	$9000
$1	$14	$2a190	$1	$800	$9800
$2	$1	$2a990	$1	$800	$a000
$2	$2	$2b190	$1	$800	$a800
$2	$3	$2b990	$1	$800	$b000
$2	$4	$2c190	$1	$800	$b800
$2	$5	$2c990	$1	$800	$c000
$2	$6	$2d190	$1	$800	$c800
$2	$7	$2d990	$1	$800	$d000
$2	$8	$6b6e0	$1	$800	$d800
$2	$9	$6bee0	$1	$800	$e000
$2	$a	$69220	$1	$150	$e800

fs_fcb

Return a pointer to the CM FCB entry which corresponds to the specified PACB pointer for a CM disc file. Non-disc files will return NMNIL.

Syntax

```
fcb_ptr:ptr = fs_fcb (pacb_ptr:ptr)
```

Parameters

fcb_ptr
The virtual address of the File Control Block is returned.

pacb_ptr
The virtual address of the pacb for the file (required).

User Discussion

The FCB is only present for disc files. If a PACB pointer is passed in w: iers to a non-disc CM file, NMNIL will be returned.

Macros Used

fs_cmcb

Example

```
$nmdat> wl fs_fcb(pacbptr)
$00000000
```

fs_file

Print detailed information about the specified file associated with a process.

Syntax

```
fs_file([pin_num:int=0],
        fnum:int,
        [detail:int=5],
        [error_parm:str='pad'])
```

Parameter

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter). This parameter is passed to the called macros (which accept a detail parameter).

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. Note that the macros are set up in such a way as to not terminated on errors caused by the File Label Table not being resident in memory at the time of the dump. These errors will be printed, and execution will continue regardless of the error_parm passed in. (optional parameter)

User Discussion

Calls the following macros (passing the detail to the appropriate macros) for the specified file:

```
fs_fname
fs_acc_attr
fs_perm
fs_dev
fs_addr
fs_disc_alc
```

Macros Used

```
fs_plfd
fs_gufd
fs_fname
fs_acc_attr
fs_perm
fs_dev
fs_addr
fs_disc_alc
```

Example

```
$nmdat> fs_file(1d,9,3)

TAPEFILE

Compatibility Mode file

Access options:          OUTPUT,NOMR,NOLOCK,EXC,BUF,NOMULTI,WAIT,NOCOPY
Access method:           $0
Last error number:       $0
File pointer:            $63
Num of multiacc users:   $0

Device type:             $0
Record type:             $0
File type:               $8

File options:            NEW,ASCII,FORMAL,V,CCTL,FEQ,STD,NOLABEL

ldev:                    $7
```

```
$nmdat> fs_file(23,9)

XCL.LIB.SYS

Native Mode file
Access options:          EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
Access method:           $0
Last error number:       $0
File pointer:            $0
Num of multiacc users:   $4
Eof pointer:             $fbf00
Total num of readers:    $4
Total num of writers:    $4

Device type:             $0
Record type:             $0
File type:               $0
File options:            SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
File code:               $408
Record size:             $100
Block size:              $100
Record limit:            $fbf00

ldev:                    $1
ufid:                    HPE_SYSTEM_VOLUME_SET:MEMBER1:$bc7

Virtual address:         $35.0
Ending virt addr:        $35.fbeff
ext blk   extent                vs        # sectors/     virt sect
   #        #       sector      index       extent      num in file
-------   ------    ------      -----     ----------    -----------
 $1        $1       $b1de0       $2          $fc0          $0
```

fs_file_perm

Print detailed information about a file, specified by file name. This macro may be very slow as the entire VSOD may need to be scanned.

Syntax

```
fs_file_perm (filename:str,
              [detail:int=5],
              [error_parm:str='pad'])
```

Parameters

filename

the filename of the file to be displayed. See the standard filename formatting convention in the file system function section for a description of how this parameter is to be formatted (required parameter).

detail

Specify a number between 0 and 5 which determines the level of detail to print (optional parameter). This parameter is passed to the called macros (which accept a detail parameter).

0	Suppress printing.
1	Print pin number.
2 - 3	Print virtual address, current job size, ext blk #, extent #, sector, VS index, # sectors/extent, virtual sector # in file, ldev #, ufid, share mode # of users this mode, file pointer.
4 - 5	Print all level 1-3 information and the device type, record type, file type, file options, filecode, record size, block size, record limit, access options, access method and last error number.

error_parm

Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Searches the VSOD/GUFD table for a match. Will only work when the file is a disc file whose file label is in memory at the time of the dump.

Macros Used

```
fs_fname_to_gufd
fs_addr
fs_disc_alc
fs_next_gdpd
fs_next_plfd
fs_dev
fs_acc_tree
```

Example

```
$nmdat> fs_file_perm('NL.PUB.SYS')

Virtual address:          $a.0
Ending virt addr:         $a.d772ff

ext blk    extent                      vs          # sectors/      virt sect
   #          #          sector        index         extent       num in file
-------    ------      ------        -----       ----------     -----------
 $1         $1         $3ac80         $1            $800          $0
 $1         $2         $3b480         $1            $800          $800
 $1         $3         $3bc80         $1            $800          $1000
 $1         $4         $3c480         $1            $800          $1800
 $1         $5         $3cc80         $1            $800          $2000
 $1         $6         $3d480         $1            $800          $2800
 $1         $7         $3dc80         $1            $800          $3000
 $1         $8         $3e480         $1            $800          $3800
 $1         $9         $3ec80         $1            $800          $4000
 $1         $a         $3f480         $1            $800          $4800
 $1         $b         $3fc80         $1            $800          $5000
 $1         $c         $40480         $1            $800          $5800
 $1         $d         $40c80         $1            $800          $6000
 $1         $e         $41480         $1            $800          $6800
 $1         $f         $41c80         $1            $800          $7000
 $1         $10        $42480         $1            $800          $7800
 $1         $11        $42c80         $1            $800          $8000
 $1         $12        $43480         $1            $800          $8800
 $1         $13        $43c80         $1            $800          $9000
 $1         $14        $44480         $1            $800          $9800
```

```
$2      $1      $44c80      $1      $800      $a000
$2      $2      $6c6e0      $1      $800      $a800
$2      $3      $6ee00      $1      $800      $b000
$2      $4      $6f600      $1      $800      $b800
$2      $5      $6fe00      $1      $800      $c000
$2      $6      $70600      $1      $800      $c800
$2      $7      $70e00      $1      $780      $d000

ldev:                       $1
ufid:                       HPE_SYSTEM_VOLUME_SET:MEMBER1:$db

Device type:                $0
Record type:                $0
File type:                  $0

File options:               SYS,BINARY,FORMAL,F,NOCCTL,FEQ,STD,NOLABEL
File code:                  $406
Record size:                $100
Block size:                 $100
Record limit:               $d77300

Share mode:                 EXCLUSIVE
Num of users this mode:     $1
File pointer:               $0

    Pin $1
    Access options:         IN/OUT,NOMR,NOLOCK,SHR,BUF,NOMUL      T,NOCOPY
    Access method:          $0
    Last error number:      $0
```

fs_file_ufid

Print detailed information about the specified permanent disc file. The file is specified by UFID. This macro may be very slow as the entire VSOD may need to be scanned.

Syntax

```
fs_file_ufid (ufid:str,
              [detail:int=5],
              [error_parm:str='pad'])
```

Parameters

ufid

unique file identification for the file to be used. See the UFID display/specification format specified above.

detail

Specify a number between 0 and 5 which determines the level of detail to print (optional parameter). This parameter is passed to the called macro (which accepts the detail parameter)

error_parm

Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Searches the VSOD/GUFD table for a match. (Calls fs_ufid_to_gufd).

Macros Used

```
fs_ufid_to_gufd
fs_addr
fs_disc_alc
fs_next_gdpd
fs_next_plfd
fs_perm
fs_dev
fs_acc_tree
```

Example

```
$nmdat> fs_file_ufid('HPE_SYSTEM_VOLUME_SET:MEMBER1:$bc7')

Virtual address:          $35.0
Ending virt addr:         $35.fbeff
```

```
ext blk   extent                    vs        # sectors/     virt sect
   #        #        sector        index        extent      num in file
-------   ------   ------         -----      ----------    -----------
 $1        $1       $b1de0         $2           $fc0          $0


Device type:                $0
Record type:                $0
File type:                  $0

File options:               SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
File code:                  $408
Record size:                $100
Block size:                 $100
Record limit:               $fbf00

ldev:                       $1
ufid:                       HPE_SYSTEM_VOLUME_SET:MEMBER1:$bc7


Device type:                $0
Record type:                $0
File type:                  $0

File options:               SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
File code:                  $408
Record size:                $100
Block size:                 $100
Record limit:               $fbf00

Share mode:                 SHARED GMULTI
Num of users this mode:     $4
File pointer:               $0
Job:                        $a

    Pin $2a  (#J8)
    Access options:         EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:          $0
    Last error number:      $0

    Pin $27  (#J10)
    Access options:         EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
    Access method:          $0
    Last error number:      $0

    Pin $24  (#J9)
    Access options:         EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
  Access method:          $0
  Last error number:      $0

  Pin $23  (#J10)
  Access options:         EXECUTE,NOMR,NOLOCK,SHR,BUF,GMULTI,WAIT,NOCOPY
     Access method:          $0
     Last error number:      $0
```

fs_filename

Print and return the filename of the specified file.

Syntax

```
filename:str = fs_filename ([pin_num:int=0],
                            fnum:int,
                           [detail:int=5],
                           [error_parm:str='pad'])
```

Parameters

filename
The filename is returned as a string formatted according to the filename formatting convention described in the introduction.

pin_num
Process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Prints out filename.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Returns and/or prints the filename associated with the specified file. The filename is returned (and printed) according to the filename specification convention outlined at the beginning of this section. Note that for NM files, the file label where the file name is stored may not be resident in memory at the time of the dump (calls fs_fname).

Macros Used

```
fs_plfd
fs_fname
```

Example

```
$nmdat> fs_filename(21,8)
SL.PUB.SYS

$nmdat> wl fs_filename(1d,a,0)
$STDLIST
```

fs_filename_ufid

Print and return the filename associated with the specified UFID. This macro may be very slow as the entire VSOD may need to be scanned.

Syntax

```
filename:str = fs_filename_ufid (ufid:str,
                                 [detail:int=5],
                                 [error_parm:str='pad'])
```

Parameters

filename
The filename is returned as a string formatted according to the filename formatting convention described in the introduction.

ufid
Specify the Unique File Identification to print the filename for.

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Enables printing.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Returns and prints the filename associated with the specified ufid. The filename is returned (and printed) according to the filename specification convention outlined at the beginning of this section. Note that the filename will not be printed if the disc file's File label was not swapped in at the time of the dump. (Calls fs_ufid_to_gufd).

Macros Used

```
fs_ufid_to_gufd
fs_fname_nm
```

Example

```
$nmdat> fs_filename_ufid('HPE_SYSTEM_VOLUME_SET:MEMBER1:$81')
SL.PUB.SYS
```

fs_flab

Return the address of the FLAB entry corresponding to the specified GUFD.

Syntax

```
flab_ptr:ptr = fs_flab (gufd_ptr:ptr,
                        [error_parm:str='pa'])
```

Parameters

flab_ptr
The virtual address of the file label that corresponds to the specified GUFD.

gufd_ptr
A pointer to the GUFD entry for the file (required parameter).

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message and add the message to the error stack. (optional parameter)

User Discussion

Returns the file label pointer by referencing the file label pointer contained in the GUFD.

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_flab(gufdptr)
$11.bc720
```

fs_fname

Return the filename associated with the specified PLFD. For NM disc files, the File Label must be in memory.

Syntax

```
filename:str = fs_fname (plfd_ptr:ptr)
```

Parameters

filename
The filename of the file that corresponds to the specified PLFD entry. The filename is in the standard filename format discussed in the file system function section.

plfd_ptr
The virtual address of the PLFD entry. (required parameter)

User Discussion

For CM files, the name is obtained from the LACB and PACB. For NM disc files, the name is obtained from the File Label (which must be in memory).

Macros Used

```
fs_is_cm_file
fs_fname_cm
fs_lacb
fs_pacb
delete_blanks
fs_fcb
fs_fname_nm
```

Example

```
$nmdat> wl fs_fname(plfdptr2)
XCL.LIB.SYS
```

fs_fname_to_gufd

Return the address of the GUFD associated with the specified file name. Note that this will be slow, since the entire VSOD will need to be scanned.

Syntax

```
gufd_ptr:ptr = fs_fname_to_gufd
                (filename:str)
```

Parameters

gufd_ptr
Returns the address of the GUFD entry.

filename
The filename to find the GUFD for. This must be in the format specified in the file system low-level macros section. (required parameter)

User Discussion

This macro obtains the GUFD pointer by searching all GUFD entries on the system. There is one GUFD for each open disc file. Since this table can be large, this macro may take a long time.

Macros Used

```
kso_pointer
kso_number
first_entry
fs_fname_nm
stderr
```

Example

```
$nmdat> wl fs_fname_to_gufd('nl.pub.sys')
$a.c1400710
```

fs_foptions

Return the foptions for the currently open file referred to by the specified PLFD.

Syntax

```
foptions:u16 = fs_foptions (plfd_ptr:ptr,
                            [flab_resident:bool=true])
```

Parameters

foptions
The foptions word as specified in the File Label for disc files, or the PACB for CM files.

plfd_ptr
A pointer to the PLFD entry for the file (required parameter).

flab_resident
A flag indicating whether the File Label is in memory or not. If this is false, the File Label will not be accessed.

User Discussion

Used to return the FOPTIONs for a given file. For CM files, this information is obtained from the PACB. For NM files, it is obtained from the File Label.

Macros Used

```
fs_is_cm_file
fs_lacb
fs_pacb
fs_gufd
fs_flab
```

Example

```
$nmdat> wl fs_foptions(plfdptr):'%'
%504
```

fs_format_aoptions

Print and return a string with the formatted aoptions.

Syntax

```
aopt_str:str = fs_format_aoptions (aoptions:u16,
                                   [column:int=1])
```

Parameters

aopt_str
A string containing the formatted aoptions is returned here.

aoptions
The aoptions value to format. (required parameter)

column
The column at which the printed string is to be left-justified.

User Discussion

The specified aoptions value is used to format and output the returned string such that it looks like the standard aoptions display from PRINTFILEINFO.

Macros Used

```
fs_faopt1
fs_faopt2
format_str
```

Example

```
$nmdat> fs_format_aoptions(fs_aoptions(plfdptr),#10)
        OUTPUT,NOMR,NOLOCK,EXC,BUF,NOMULTI,WAIT,NOCOPY
```

fs_format_foptions

Print and return a string with the formatted foptions.

Syntax

```
fopt_str:str = fs_format_foptions (foptions:u16,
                                   [column:int=1])
```

Parameters

fopt_str
A string containing the formatted foptions is returned.

foptions
The foptions value to format. (required parameter)

column
The place at which the printed string is to be left-justified.

User Discussion:

The specified foptions value is used to format and output the returned string such that it looks like the standard aoptions display from PRINTFILEINFO.

Macros Used

```
fs_1_format_foptions
format_str
```

Example

```
$nmdat> fs_format_foptions(fs_foptions(plfdptr),#20)
                    NEW,ASCII,FORMAL,V,CCTL,FEQ,ST ,NOLABEL
```

fs_ftype

Return the file type of the file referred to by the PLFD.

Syntax

```
ftype:int = fs_ftype (plfd_ptr:ptr)
```

Parameters

ftype
The file type value.

plfd_ptr
The virtual address of the PLFD. (required parameter)

User Discussion

Returns the file type value from the PLFD.

Macros Used

none

Example

```
$nmdat> wl fs_ftype(plfdptr)
$8
```

fs_gufd

Return the address of the GUFD corresponding to the specified PLFD.

Syntax

```
gufd_ptr:ptr = fs_gufd (plfd_ptr:ptr)
```

Parameters

gufd_ptr
The virtual address of the GUFD is returned.

plfd_ptr
Specify the virtual address of the PLFD (required parameter).

User Discussion

Returns NMNIL if the PLFD has no GUFD entry ($NULL or non-disc).

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_gufd(plfdptr)
$00000000

$nmdat> wl fs_gufd(plfdptr2)
$c14011d0

$nmdat> var gufdptr2 = fs_gufd(fs_plfd(23,d))
```

fs_is_cm_file

Return TRUE if the specified file is a Compatibility Mode file type.

Syntax

```
ret_val:bool = fs_is_cm_file (plfd_ptr:ptr)
```

Parameters

ret_val
This will be returned TRUE for all CM files.

plfd_ptr
The virtual address of the PLFD entry for the file. (required parameter)

User Discussion

This macro examines the file type field in the PLFD.

Macros Used

none

Example

```
$nmdat> wl fs_is_cm_file(plfdptr)
TRUE
```

fs_lacb

Return a pointer to the CM LACB entry which corresponds to the specified PLFD pointer for a CM file. Will return NMNIL for non-multi access files.

Syntax

```
lacb_ptr:ptr = fs_lacb (plfd_ptr:ptr)
```

Parameters

lacb_ptr
The virtual address of the Logical Access Control Block is returned.

plfd_ptr
The virtual address of the plfd for the file (required).

Macros Used

```
objcl_number
stderr
fs_aft
fs_cmcb
```

Example

```
$nmdat> wl fs_lacb(fs_plfd(23,d))
$f8.40011e3c
```

fs_ldev

Return the logical device number of the file referred to by the specified PLFD.

Syntax

```
ldev:int = fs_ldev (plfd_ptr:ptr)
```

Parameters

ldev
The logical device number that the specified file resides on.

plfd_ptr
A pointer to the PLFD entry for the file (required parameter).

User Discussion

For CM files, the ldev is found in the PACB. For NM files, the ldev is found in the MVT entry for the file, and will be the device that the File Label resides on (since the file can be spread over multiple volumes).

Macros Used

```
fs_gufd
fs_is_cm_file
fs_pacb
fs_mvt
```

Example

```
$nmdat> wl fs_ldev(plfdptr)
$7
```

fs_mounted_volumes

Print information about, and return a list of pointers to the Mounted Volume Table entries of all volumes currently mounted on the system.

Syntax

```
vol_list:str = fs_mounted_volumes ([detail:int=5])
```

Parameter

vol_list
A list of pointers suitable for use with the DAT FOREACH command. Each pointer points to a Mounted Volume Table entry.

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Print out volume set name and volume name.

User Discussion

Runs through the Mounted Volume Table.

Macros Used

```
kso_pointer
kso_number
delete_blanks
```

Example

```
$nmdat> wl fs_mounted_volumes(0)
 $c4070900 $c4070b00

$nmdat> fs_mounted_volumes()

Volume set name          Volume name
---------------          -----------
HPE_SYSTEM_VOLUME_SET     MEMBER1
HPE_SYSTEM_VOLUME_SET     MEMBER2
```

fs_mvt

Return the virtual address of the Mounted Volume Table entry for the specified GUFD.

Syntax

```
mvt_ptr:ptr = fs_mvt (gufd_ptr:ptr)
```

Parameters

mvt_ptr
The virtual address of the MVT entry that corresponds to the specified volume id.

gufd_ptr
A pointer to the GUFD entry for the file (required parameter).

User Discussion

Uses the UFID field of the GUFD entry to index into the MVT.

Macros Used

```
kso_pointer
kso_number
stderr
```

Example

```
$nmdat> wl fs_mvt(gufdptr)
$c4070900
```

fs_next_aft

Return the MPE V file number of the next CM active file table entry (AFT) for the specified pin.

Syntax

```
aft_ptr:ptr = fs_next_aft
                (pin_num:int,
                [file_num:int=0])
```

aft_ptr
The virtual address of the AFT entry found. Will return zero if no higher open file exists for the pin.

pin_num
process id for the AFT to be examined (required).

file_num
CM file number indicating an open file for the given pin (optional).

User Discussion

If a filenum is specified, it will return the next higher valid entry. If there is no open filenum with a higher number, 0 will be returned. If no filenum is specified, a pointer to the first entry will be returned.

Macros Used

```
fs_aftcm
stderr
```

Example

```
$nmdat> wl fs_next_aft(1d)
$1

$nmdat> wl fs_next_aft(1d,9)
$a
```

fs_next_fnum

Return the file number of the next file in file number sequence for the specified pin.

Syntax

```
return_fnum:int = fs_next_fnum ([pin_num:int=pin],
                                [fnum:int=-1])
```

Parameters

return_fnum
The next NM file number in sequence is returned here. -1 is returned if there are no more valid files.

pin_num
process id. If omitted, the current process process id will be used. (optional parameter).

fnum
file number of the file within the specified process. Omit this parameter if the first file number is desired. (optional parameter).

User Discussion

Returns the next NM file number in sequence (the first number will be zero). Unallocated files are skipped, and -1 is returned after the last valid file number.

Macros Used

```
fs_plfd
stderr
```

Example

```
$nmdat> wl fs_next_fnum(1d,9)
$a
```

fs_next_gdpd

Return the address of the next GDPD for the specified GUFD and GDPD.

Syntax

```
return_gdpd_ptr:ptr = fs_next_gdpd (gufd_ptr:ptr,
                                    [gdpd_ptr:ptr=NMNIL])
```

Parameters

return_gdpd_ptr
The virtual address of the next GDPD in the chain (starting from the specified GUFD) is returned here. NMNIL is returned if there is no next GDPD.

gufd_ptr
Specify the virtual address of the GUFD (required parameter).

gdpd_ptr
Specify the virtual address of the current GDPD entry. The entry in the chain after this entry will be returned. If this is not specified, the first entry will be returned. (optional parameter)

user disc
This will only work for files which are shared through NM structures (GDPD).

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_next_gdpd(gufdptr)
$820032d4
```

fs_pacb

Return a pointer to the CM PACB entry which corresponds to the specified PLFD pointer for a CM file.

Syntax

```
pacb_ptr:ptr = fs_pacb (plfd_ptr:ptr)
```

Parameters

pacb_ptr
The virtual address of the Physical Access Control Block is returned.

plfd_ptr
The virtual address of the plfd for the file (required).

Macros Used

```
objcl_number
stderr
fs_aft
fs_cmcb
```

Example

```
$nmdat> wl fs_pacb(plfdptr)
$b.83360020

$nmdat> var pacbptr = fs_pacb(plfdptr)
```

fs_perm_char

Print the permanent characteristics of the specified file. Return the foptions for this file.

Syntax

```
foptions:u16 = fs_perm_char ([pin_num:int=0],
                              fnum:int,
                             [detail:int=5],
                             [error_parm:str='pad'])
```

Parameters

foptions
The file options for the file are returned.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 2	Print the foptions only.
3 - 5	Print foptions and some perm chars.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Displays all permanent characteristics associated with the specified file. Environment specific characteristics may also appear. (Calls fs_perm).

Macros Used

```
fs_plfd
fs_perm
```

Example

```
$nmdat> fs_perm_char(23,d)

Device type:            $0
Record type:            $0
File type:              $8
File options:           JOB,ASCII,FORMAL,V,NOCCTL,FEQ,MSG,NOLABEL
Record size:            $100
Block size:             $85
Record limit:           $407

$nmdat> fs_perm_char(1d,a,3)

Device type:            $0
Record type:            $0
File type:              $8

File options:           ALL,ASCII,$STDLIST,U,CCTL,FEQ,STD,NOLABEL
```

fs_plfd

Return the address of the PLFD corresponding to the specified file.

Syntax

```
plfd_adr:ptr = fs_plfd ([pin_num:int=pin],
                        fnum:int)
```

Parameters

plfd_adr
The virtual address of the PLFD is returned here.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The NM file number of the file within the specified process (required parameter).

User Discussion:

The address of the PLFD which corresponds to the specified file number and pin is returned.

Macros used

```
pm_kpo_pointer
pm_kpo_number
stderr
tbl_entry_allocated
```

Example

```
$nmdat> wl fs_plfd(1d,9)
$b.83383140
```

fs_pxfile

Return a pointer to the PXFILE area of the CM stack given a pointer to the beginning of the stack.

Syntax

```
pxfile_ptr:ptr = fs_pxfile (stack_ptr:ptr)
```

Parameters

pxfile_ptr
The virtual address of the start of the CM PXFILE area is returned.

stack_ptr
The virtual address of the CM data segment stack (required).

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_pxfile(cmstkptr)
$fd.40011d68
```

fs_scan_fmavt

Return a pointer to the FMAVT entry corresponding to the file given by the GUFD pointer.

Syntax

```
fmavt_ptr:ptr = fs_scan_fmavt (gufd_ptr:ptr)
```

fmavt_ptr
The virtual address of the FMAVT entry for the given file is returned. If the file is not opened CM multi-access, there will be no entry, and zero will be returned.

gufd_ptr
The virtual address of the gufd for a file (required).

Macros Used

none

Example

```
$nmdat> wl fs_scan_fmavt(gufdptr2)
$00900008
```

fs_ufid

Print and return the Unique File ID of the specified file.

Syntax

```
ufid:str = fs_ufid ([pin_num:int=0],
                    fnum:int,
                    [detail:int=5],
                    [error_parm:str='pad'])
```

Parameters

ufid
The UFID is returned here in the format specified in the formatting conventions section.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

fnum
The MPE/XL file number of the file within the specified process (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print (optional parameter).

0	Suppresses printing.
1 - 5	Enables printing.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Returns and prints the UFID associated with the specified file (calls fs_ufid_str).

Macros Used

```
fs_plfd
fs_gufd
fs_ufid_str
```

Example

```
$nmdat> fs_ufid(21,8)
HPE_SYSTEM_VOLUME_SET:MEMBER1:$81

$nmdat> wl fs_ufid(23,d,0)
HPE_SYSTEM_VOLUME_SET:MEMBER1:$b40
```

fs_ufid_file

Print and return the UFID associated with the specified file name. This macro may be very slow as the entire VSOD may need to be scanned.

Syntax

```
ufid:str = fs_ufid_file (filename:str,
                         [detail:int=5],
                         [error_parm:str='pad'])
```

Parameters

ufid
the UFID is returned in its internal representation (optional parameter).

filename
the filename of the file to be displayed. See the standard filename formatting convention above for a description of how this parameter is to be formatted (required parameter).

detail
Specify a number between 0 and 5 which determines the level of d print (optional parameter).

0	Suppresses printing.
1 - 5	Print system volume set .

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Searches the VSOD/GUFD table for a match. (Calls fs_ufid_to_gufd).

Macros Used

```
fs_fname_to_gufd
fs_ufid_str
```

Example

```
$nmdat> fs_ufid_file('nl.pub.sys')
HPE_SYSTEM_VOLUME_SET:MEMBER1:$db
```

fs_ufid_str

Return the UFID corresponding to the specified GUFD.

Syntax

```
ufid:str = fs_ufid_str (gufd_ptr:ptr)
```

Parameters

ufid
The UFID is returned here in the external display format as specified in the file system function section. (required parameter)

gufd_ptr
The virtual address of the GUFD entry for the file. (required parameter)

User Discussion

Grabs the needed info from the MVT.

Macros Used

```
fs_mvt
delete_blanks
fs_vol_set_name
fs_vol_name
```

Example

```
$nmdat> wl fs_ufid_str(gufdptr)
HPE_SYSTEM_VOLUME_SET:MEMBER1:$bc7
```

fs_ufid_to_gufd

Return the address of the GUFD corresponding to the specified UFID. This macro will be very slow as the entire VSOD may need to be scanned.

Syntax

```
gufd_ptr:ptr = fs_ufid_to_gufd (ufid:str)
```

Parameters

gufd_ptr
The virtual address of the GUFD that corresponds to the specified UFID.

ufid
The UFID to get the gufd pointer for. Specified in the external display format as documented in the file system function section. (required parameter)

User Discussion

Runs thru the VSOD/GUFD table looking for a UFID match. May take a long time.

Macros Used

```
kso_pointer
kso_number
first_entry
fs_ufid_str
stderr
```

Example

```
$nmdat> wl fs_ufid_to_gufd('HPE_SYSTEM_VOLUME)
$a.c14011d0
```

fs_vol_name

Return the volume name for the specified Mounted Volume Table entry.

Syntax

```
volname:str = fs_vol_name (mvt_ptr:ptr)
```

volname The volume name of the specified volume is returned here.

mvt_ptr
The virtual address of the Mounted Volume Table entry. (required parameter)

User Discussion

The volume name is grabbed from the MVT entry.

Macros Used

none

Example

```
$nmdat> wl fs_vol_name(mvtptr)
MEMBER1
```

fs_vol_set_name

Return the volume set name for the specified Mounted Volume Table entry.

Syntax

```
setname:str = fs_vol_set_name (mvt_ptr:ptr)
```

setname
The volume set name of the specified volume is returned here.

mvt_ptr
The virtual address of the Mounted Volume Table entry. (required parameter)

User Discussion

The volume set name is grabbed from the MVT entry.

Macros Used

none

Example

```
$nmdat> wl fs_vol_set_name(mvtptr)
HPE_SYSTEM_VOLUME_SET
```

fs_vsod

Return the address of the VSOD entry corresponding to the specified GUFD.

Syntax

```
vsod_ptr:ptr = fs_get_vsod (gufd_ptr:ptr)
```

Parameters

vsod_ptr
The virtual address of the VSOD associated with the specified GUFD.

gufd_ptr
The virtual address of the GUFD entry. (required parameter)

User Discussion

Merely subtracts an offset to generate a pointer into the same VSOD/GUFD table.

Macros Used

```
objcl_number
stderr
```

Example

```
$nmdat> wl fs_vsod(gufdptr)
$c1401178
```

fs_vtab_entry

Return a pointer to the CM file Control Block Table's Vector table entry given the CM segment-offset vector from the AFT.

Syntax

```
vtab_entry_ptr:ptr = fs_vtab_entry
                (cbtdst:int,
                cbtoff:int)
```

Parameters

vtab_entry_ptr
The virtual address of the entry in the vector table for the given vector.

cbtdst
The CM data segment number containing the control block, as given in the AFT entry (required).

cbtoff
The CM word offset into the control block table for the entry as given in the AFT (required).

Macros Used

fs_cmcbt

Example

```
$nmdat> wl fs_vtab_entry(%160,%10)
$b.83360010
```

Contents

Chapter 6
I/O Macros

6

I/O Macros

Introduction

Most I/O macros will display/return information concerning native mode or compatability mode I/O by passing logical device number or path id. The rest of the macros require addresses that can be obtained using other I/O macros.

Macros handle the following functions:

High-level input/output
Low-level input/output
Device allocation (compatability mode)

The following abbreviations are used:

CAM - Channel Adapter Manager
DA - Device Adapter
DAM - Device Adapter Manager
DCT - Device Class Table (CM)
DM - Device Manager
DMA - Direct Memory Access
EIR - External Interrupt Register
EIT - External Interrupt Table
FLIH - First Level Interrupt Handler
HLIO - High-level Input/Output
IOX - Compatability mode I/O Identifier
LDEV - Logical Device Number
LDM - Logical Device Manager
LDT - Logical Device Table (CM)
LLIO - Low-level Input/Output
LPDT - Logical/Physical Device Table (CM)
LPT - Logical Path Table
MIB - Memory Interface Block
PDA - Port Data Area
PPT - Physical Path Table
RIB - Request Information Block (CAM Data Structure)
RID - Request ID (Native mode logical I/O identifier)
SLIH - Second Level Interrupt Handler

A path is specified as:

'n.n.n...n'

Where n is the id number of the channel, device adapter, device, etc. The pathname may be up to 32 characters and must contain no embedded blanks. The following managers are supported:

Channel Adapter Manager (CAM)
HP-IB Device Adapter Manager (DAM)
TMUX Device Adapter Manager (DAM)
CS80 Disc Device Manager (DM)
7978 Tape Device Manager (DM)
Ciper Device Manager (DM)
Page Printer Device Manager (DM)
Terminal Device Manager (DM)

io_cm_ioreq

Print the CM I/O request entry associated with the specified IOX.

Syntax

```
return_val:any = io_cm_ioreq (iox:int,
                              [detail:int=1],
                              [field_name:str=''])
```

Parameters

return_val
If the field_name is specified and a single entry is selected, the value of the field is returned here.

iox
The CM I/O transaction number is used to find the I/O request entry. (required parameter)

detail

0	Suppress printing.
1 - 5	Causes the specified field (or all fields) to print.

The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Used to get a specified I/O request issued from compatability mode using the CM IO identifier (IOX).

The CM IO request table is defined in DIOCM.HLIO. The pointer to the table header is kso_hlio_cmio_hdr. Scan through the table (using get_entry_ptr) until the specified IOX matches the cmio_iox field in the table.

If the table entry is not found, cause an error. When the table entry is found:

Macros Used

```
kso_pointer
kso_number
get_entry_ptr
```

Example

```
$2eb ($36) nmdat >io_cm_ioreq(1)

PACKED RECORD
   CMIO_IOX : 1
   REQ_PTR  : 806201d0
END
```

io_cmdevice_class

Returns a string with a list of the CM device classes associated with the specified LDEV.

Syntax

```
class_list:str = io_cmdevice_class (ldev:int)
```

Parameters

class_list
A string containing each of the device classes associated with the specified logical device separated by a space.

ldev
The logical device number of the device to return the class information for. (required parameter)

User Discussion

Get the DCT and read each class entry searching for the specified LDEV - The DCT is in the same place and format as MPE V - see the MPE V tables manual.

Macros Used

Example

io_config_ldev

Print the configuration information for the specified LDEV.

Syntax

```
io_config_ldev (ldev:int,
                [detail:int=5]
                [header:bool = true])
```

Parameters

ldev

The logical device number of the device to print the configuration information for.. (required parameter)

detail

0 - 2 Print ldev # and device type.

3 - 5 Print level 0 - 2 info plus I/O path, id, outdev, mode, size and class.

The default is 5.(optional parameter)

header

If true then header for the information will be printed. (optional parameter)
The default is true.

User Discussion

Use io_lpt to get the logical path table information.

Use io_ldev_to_path to get the path spec for the device.

Use io_ppt to get the physical configuration information.

Print all of the above according to the example.

Macros Used

```
io_lpt
io_ldev_to_path
io_ppt
```

Example

```
$2ed ($36) nmdat > io_config_ldev(7)

LDEV# IO-PATH  TYPE     ID       OUTDEV        MODE     SIZE       CLASS
----- -------  ----     --       ------        ----     ----       -----
$7    2/4.2.3  IO_TAPE  HP7978   $320                   $0         TAPE
                                                                   BUCKHORN

$2ee ($36) nmdat > io_config_ldev(7,2,FALSE)

$7               IO_TAPE
```

io_config_path

Print the configuration information for the specified path specification.

Syntax

```
io_config_path (path:int,
                [detail:int=5]
                [header:bool=true])
```

Parameters

path

The path specification of the device to print the configuration information for. (required parameter)

detail

1 - 3 Print path and device type.
4 - 5 Print level 0 - 2 information plus ldev, id, outdev, move, size and class.

The default is 5.(optional parameter)

header

If true then a header for the information will be printed. (optional parameter)
The default is true.

User Discussion

Use io_ppt to get the physical configuration information. Print it according to the example.

Macros Used

io_ppt

Example.

```
$2ef ($36) nmdat > io_config_path('2/4.2.3')

LDEV# IO-PATH  TYPE        ID       OUTDEV    MODE      SIZE      CLASS
----- -------  ----        --       ------    ----      ----      -----
$7    2/4.2.3  IO_TAPE     HP7978   $320                $0        TAPE
                                                                   BUCKHORN


$2f0 ($36) nmdat > io_config_path('2/4.2.3',1,FALSE)

2/4.2.IO_TAPE
```

io_data_chain

Print a data chain record.

Syntax

```
return_val:any = io_data_chain (address:ptr,
                                [selector:str=''],
                                [detail:int=1])
```

Parameters

return_val
The value of the specified selector is returned here only if a selector is specified.

address
The virtual address of the data chain entry to print must be specified here. (required parameter)

selector
Specify a one of the following selectors to cause the value of that selector to be returned and printed:

address	The virtual address of the data chain entry.
count	The number of bytes to transfer for this data chain.

If the selector is omitted, nothing will be returned, and both fields will be printed. (optional parameter)

detail
Specify 0 to suppress printing;

1 - 5 Detailed information with heading.

The default is 1. (optional parameter)

User Discussion

Macros Used

io_ppt

Example

```
$2e8 ($36) nmdat > io_data_chain(b.80120048)

BUFF POINTER     LENGTH
------------     ------
$11c.40200140    $100
```

io_dct

Print the Device Class Table (DCT) entry associated with the specified device class.

Syntax

```
return_val:any = io_dct (device_class:str,
                         [detail:int=1],
                         [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

device_class
Specify the device class to print the table entry for. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. Specify 'ldev' to return a list of all of the ldevs associated with the device class. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Since the device class table is a CM table, refer to the MPE V tables manual (13-15) for details on how to access and read it.

```
pointer = address of device class entry.
if field_name = 'pointer'
return pointer
```

Print the entry out, and return the specified field, if specified. If 'ldev' is specified, print all of the ldevs and return a list of the ldevs associated with the class.

Macros Used

none

Example

```
$3e6 ($0) nmdat >io_dct('disc')

CRUNCHED RECORD
   CLASS_NAME        : 'DISC     '
   FILLER_1          : FALSE
   CYCLICAL_POINTER  : 1
   SPOOLED           : FALSE
   TERMINAL          : FALSE
   CLASS_TYPE        : 3
   DEVICE_CNT        : 8
END

$2f1 ($36) nmdat >io_dct('tape')

CRUNCHED RECORD
   CLASS_NAME        : 'TAPE     '
   FILLER_1          : FALSE
   CYCLICAL_POINTER  : 1
   SPOOLED           : FALSE
   TERMINAL          : FALSE
   CLASS_TYPE        : 18
   DEVICE_CNT        : 1
END

ldev: $7
```

io_docket

Prints the docket table for the IOLDM for the specified LDEV.

Syntax

```
return_val:any = io_docket (ldev:int,
                            [entry_num:int=0],
                            [detail:int=1],
                            [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
Specify the logical device number of the docket table to print. This device must have an IO/LDM associated with it or unpredictable results will occur. (required parameter)

entry_num
Specify an entry number within the docket table to print. One is the first entry. The default is print all entries. (optional parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print.

The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

- The format of the docket array is contained in TIOLDM.HLIO.
- The docket array is in the field 'docket' in the
- type ldm_server_data_area.
- The type of each docket entry is active_docket.

Print the entry (or all docket entries) out, and return the specified field, if specified.

Macros Used

```
io_lpt
port_data
```

Example

```
$2f3 ($36) nmdat >io_docket(3)

[ 1 ]:
   THIS_DOCKET       : 0
   IOQ_PTR           : 0
   START_TIME        : 00
   FREEZE_MASK       : [ ]
   FREE              : TRUE
   ABORT_PENDING     : FALSE
   SWAPIN_PENDING    : FALSE
   BROKEN_READ       : FALSE
   ALREADY_NOTIFIED  : FALSE
   REQUEUE           : FALSE
   RTN_GOOD_STATUS   : FALSE
[ 2 ]:
   ...
[ 3 ]:
   ...
[ 4 ]:
   ...
[ 5 ]:
   ...
[ 6 ]:
   ...
[ 7 ]:
   THIS_DOCKET       : 0
   IOQ_PTR           : 0
   START_TIME        : 00
   FREEZE_MASK       : [ ]
   FREE              : TRUE
   ABORT_PENDING     : FALSE
   SWAPIN_PENDING    : FALSE
   BROKEN_READ       : FALSE
   ALREADY_NOTIFIED  : FALSE
   REQUEUE           : FALSE
   RTN_GOOD_STATUS   : FALSE
```

io_errors

Prints the last few i/o errors that have occurred.

Syntax

```
io_errors ([detail: int = 5])
```

Parameters

detail

0	Suppress printing.
1 - 5	Print the i/o error message.

The default is 5.(optional parameter)

Example

```
$405 ($0) nmdat > io_errors

*****************          *************************
OS STATUS VALUES             IO SERVER STATUS VALUES
*****************          *************************

Info   Subsys                Error num Proc num  Subsys
----   ------                --------- --------  ------
$600   $81                   $6        $ffb4     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $26       $fff7     IO Services
$0     $0                    $50       $ffdc     IO Services
$0     $0                    $0        $0        $0
$0     $0                    $0        $0        $0
$0     $0                    $0        $0        $0
$0     $0                    $0        $0        $0
$0     $0                    $0        $0        $0
```

io_hlio_global

Print the HLIO Global area (Sanctum).

Syntax

```
return_val:any = io_hlio_global ([detail:int=1],
                                 [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Macros Used

kso_pointer
kso_number

Example

```
$2f5 ($36) nmdat >io_hlio_global

RECORD
   HLIO_MI                 : FALSE
   HLIO_DEBUG              : FALSE
   LDEV_LDM_DEBUG          : 7fff
   DEBUG_OPTION            : [ ]
   SIGNAL_MONITOR          : 0
    .....
    .....
      [ 23 ]:
         TRACE_ID : IOT_FIRST
         LDEV     : 0
         RID      : 0
         PORT     : 0
         MISC     : 0
END
```

io_ioldm_port

Prints the port data area for the IO/LDM port for the specified LDEV.

Syntax

```
return_val:any = io_ioldm_port (ldev:int,
                                [detail:int=1],
                                [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
Specify the logical device number of the device to print the IO/LDM for. This device must utilize an IO/LDM otherwise invalid results will occur. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

The docket field needs to be suppressed, since this will result in a huge printout. Since ranges are not implemented in DAT, each field in the LDM must be specified individually. The code below will not apply as is -- it will have to be changed for the above. The LDM PDA is TIOLDM.HLIO.

Macros Used

```
io_lpt
port_data
```

Example

```
$2e9 ($36) nmdat > io_ioldm_port(7)

          LDM_ID: A                          FREEZE_DATA: TRUE
       ALIGNMENT: TRUE                     DEVICE_CLASS: IO_TAPE
       THIS_LDEV: $7                       PDM_REV_CODE: $1
        PDM_PORT: $ffffffca                PDM_LOW_SUBQ: $4
    MM_SURROGATE: $ffffffda                 CONFIG_PORT: $ffffffd9
  CONFIG_SUBQUEUE: $0                     MAX_STREAMING: $1
    REQUEST_LIMIT: $1                  OUTSTANDING_REQS: $0
       SWAP_COUNT: $0                       DOCKET_LAST: $7
     DOCKET_COUNT: $7                     CONSOLE_MODE: FALSE
  AVR_IN_PROGRESS: FALSE                     LAST_TIME: $0
        TRACE_NR: $3                            LOG_NR: $0

LDM_STATE:
---------

[ LDM_BOUND_TO_DM , LDM_ENABLED ]

SIGNAL_TABLE:
------------

[ 0 ]:
   TARGET_PORT : 0
   TARGET_SUBQ : 0
   ENABLED     : FALSE
[ 1 ]:
       .....              ...
       .....              ...
       .....              ...
```

io_ios_diag_log

Logs the entry to internal diagnostic error log.

Syntax

```
io_ios_diag_log
          ([entry_num : INT / S16 = $ffff] ,
           [detail    : INT / U16 = $1])
```

Parameters

entry_num
The entry number the user wishes to print. If -1 or no entry number is passed all entries are printed.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 2 Will print the Entry #, Transaction #, From Port, Status, Retry Count, and Logging Subsystem.
3 - 5 Will do a 'FV' of the entries.

User Discussion

Whenever an event is logged by the LLIO subsystem to the system log files, IO Services will first log the entry to it's internal diagnostic error log before passing the message on to the system logging subsystem. This table holds a short list of the most recent IO events logged.

Example

```
$27c ($0) nmdat > io_ios_diag_log
```

ENTRY #	TRANS #	FROM PORT	STATUS	RETRY COUNT	LOGGING SUBSYS
$0	$626	$ffffffcd	$dfe00079	$0	$79
$1	$629	$ffffffcd	$dfe00079	$0	$79
$2	$62b	$ffffffaa	$dfe00079	$0	$79
$3	$62e	$ffffffaa	$dfe00079	$0	$79
$4	$744	$ffffffa8	$dfe00079	$0	$79
$5	$747	$ffffffa8	$dfe00079	$0	$79
$6	$0	$ffffff75	$4ff0080	$0	$80
$7	$0	$ffffff76	$4e00b4	$0	$80
$8	$0	$ffffff76	$4e00b4	$0	$80
$9	$0	$ffffff76	$4e00b4	$0	$80
$a	$0	$ffffff76	$4e00b4	$0	$80
$b	$0	$ffffff76	$4e00b4	$0	$80

```
$40e ($0) nmdat > io_ios_diag_log(,3)

              IO SERVICES DIAGNOSTIC LOGGING TABLE
              ------------------------------------

PACKED RECORD
   MSG_HEADER :
      MSG_DESCRIPTOR  : 16
      MESSAGE_ID      : 0
      TRANSACTION_NUM : 626
      FROM_PORT       : ffffffcd
   DIAG_EVENT :
      LLIO_STATUS     :
            IS_OK : dfe00079
      DIAG_CLASS      : 1
      HW_STATUS_LEN   : 14
      MGR_INFO_LEN    : 34
      RETRY_COUNT     : 0
      LOG_ALL_RETRIES : FALSE
      RETRY_AGAIN     : FALSE
      IO_WORKED       : FALSE
       ....             ...
       ....             ...
```

io_ldev_owner

Returns the pin that own the specified ldev.

Syntax

```
pin:int = io_ldev_owner (ldev:int)
```

Parameters

ldev
The logical device number of the device to check. (required parameter)

User Discussion

This macro will return the pin number of process that owns the specified ldev. Note that only CI processes will actually own the ldev even though a son of a CI also logically owns the device.

Example

```
$356 ($36) nmdat > wl io_ldev_owner(7)
$0
```

io_ldev_state

Print the LDEV state table entry associated with the specified LDEV.

Syntax

```
return_val:any = io_ldev_state (ldev:int,
                                [detail:int=1],
                                [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
The logical device number of the LDEV state table entry to print. (required parameter)

detail
Specify 0 to supress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Once found, simply index into the table by the LDEV to get the pointer to the table entry and do the following:

Macros Used

```
kso_pointer
kso_number
```

Example

```
$41f ($0) nmdat >io_ldev_state(#20)

PACKED RECORD
   OPEN_COUNT     : 7
   S_EOR_CHAR     : ''
   S_ATTN_CHAR    : ''
   INHIBIT_CRLF   : FALSE
   V3000_MODE     : FALSE
   BLOCK_MODE     : FALSE
   BINARY_MODE    : FALSE
   PRINT_ON_PERF  : FALSE
   PRESPACE       : FALSE
   FEATURE_MODE   : FALSE
END
```

io_ldev_to_path

Returns the path specification for the specified logical device number.

Syntax

```
path_spec:str = io_ldev_to_path(n)
```

Parameters

path_spec
The path specification for the specified logical device number is returned here in the standard format.

ldev
The logical device number of the device to return the path specification for. (required parameter)

User Discussion

Get the DM's portid using io_lpt (with the specified ldev). Use io_portid to get the address of the ppt record associated with the DM's portid.

The field per_path in the ppt is the path spec (use io_ppt to get it); return it.

Macros Used

```
io_lpt
io_portid
io_ppt
```

Example

```
$35a ($36) nmdat > wl io_ldev_to_path(7)

2/4.2.3
```

io_ldt

Prints the CM LDT entry for the specified LDEV.

Syntax

```
return_val:any = io_ldt (ldev:int,
                         [detail:int=1],
                         [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
The logical device number of the LDT entry to print. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

- the number of entries in the LDT is in the first 16 bits
- of the DST.

Once found, simply index into the table by the LDEV to get the pointer to the table entry and do the following:

Macros Used

Example

```
$35b ($36) nmdat >io_ldt(7)

CRUNCHED RECORD
   FILE_USE_CNT          : 0
   VOL_NDX_OR_PROC_PIN   : 0
   REC_WIDTH             : 0
   COMM_SYS              : FALSE
   FORMS                 : FALSE
   DEV_INFORMATION       :
         DEV_TYPE : 0
   SPOOL_STATE           : 0
   SYS_STATE             : FALSE
   DIAG_STATE            : FALSE
   DOWN_REQUEST          : FALSE
   TRAILERS              : FALSE
   HEADERS               : FALSE
   DEV_CLASS             : FALSE
   SPOOL_QUEUE           : FALSE
   DEV_INFO              : 0
   LDT_RESERVED_1        : 0
   XDD_HEAD_INDEX        : 0
   CNTRL_Y_PIN           : 80
   OUT_DEV_OR_CLASS_NDX  : 320
END
```

```
$35c ($36) nmdat >io_ldt(7,1)

CRUNCHED RECORD
   FILE_USE_CNT          : 0
   VOL_NDX_OR_PROC_PIN   : 0
   REC_WIDTH             : 0
   COMM_SYS              : FALSE
   FORMS                 : FALSE
   DEV_INFORMATION       :
         DEV_TYPE : 0
   SPOOL_STATE           : 0
   SYS_STATE             : FALSE
   DIAG_STATE            : FALSE
   DOWN_REQUEST          : FALSE
   TRAILERS              : FALSE
   HEADERS               : FALSE
   DEV_CLASS             : FALSE
   SPOOL_QUEUE           : FALSE
   DEV_INFO              : 0
   LDT_RESERVED_1        : 0
   XDD_HEAD_INDEX        : 0
   CNTRL_Y_PIN           : 80
   OUT_DEV_OR_CLASS_NDX  : 320
END
```

io_lpdt

Prints the CM LPDT entry for the specified LDEV.

Syntax

```
return_val:any = io_lpdt (ldev:int,
                          [detail:int=1],
                          [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
The logical device number of the LPDT entry to print. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

- the number of entries in the LPDT is in the first 16 bits
- if the DST.

Once found, simply index into the table by the LDEV to get the pointer to the table entry and do the following:

Macros Used

Example

```
$35d ($36) nmdat >io_lpdt(7)

CRUNCHED RECORD
   DEV_KIND           : TRUE
   LPDT_RESERVED_1    : 7fff
   DEVICE_OWNERSHIP   : 0
   JOB                : FALSE
   DATA               : FALSE
   CONTROL_Y          : FALSE
   DUPLICATIVE        : FALSE
   INTERACTIVE        : FALSE
   EOF_CONDITION      : 0
   BREAK              : FALSE
   LOGGING            : FALSE
   DEV_SUBTYPE        : 0
   SYSDB_DIT_POINTER  : 0
   LPDT_RESERVED_2    : 8
END
```

io_lpt

Prints the LPT entry for the specified LDEV.

Syntax

```
return_val:any = io_lpt (ldev:int,
                         [detail:int=1],
                         [field_name:str=''])
```

Parameters

return_val
If the field_name is specified, the value of the field is returned here.

ldev
The logical device number of the LPT entry to print. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Once found, simply index into the table by the LDEV to get the pointer to the table entry and do the following:

Macros Used

Example

```
$35e ($36) nmdat >io_lpt(7)

PACKED RECORD
   LDM_PORT          : ffffffc9
   DM_PORT           : ffffffca
   DEV_CLASS         : IO_TAPE
   DEV_SUBCLASS      : 1
   HW_CLASS          : 0
   FREEZE_PREP       : TRUE
   CHECK_ALIGNMENT   : TRUE
   DISC_RPS          : FALSE
   DISC_LOG          : FALSE
   CONFIG_SUBQ       : 4
   ABORT_SUBQ        : 4
   NORMALIO_SUBQ     : 7
   PREEMPT_SUBQ      : 5
   CONSOLE_SUBQ      : 2
   DM_LOW_SUBQ       : 4
   DM_HI_SUBQ        : 4
   DM_CONSOLE_SUBQ   : 4
   DATAP             : a.c0438200
   DM_DATAP          : b.801c8100
   LPT_BK_BREAK      : [ ]
   LPT_CY_BREAK      : [ ]
   LPT_BK_HLIO       : [ ]
   LPT_CY_HLIO       : [ ]
END
```

io_mib

Prints the specified MIB entry.

Syntax

```
return_val:any = io_mib (address:ptr,
                         [detail:int=1],
                         [field_name:str=''])
```

Parameters

return_val If the field_name is specified, the value of the field is returned here.

address
Specify the virtual address of the MIB entry to print. (required parameter)

detail
Specify 0 to suppress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

Macros Used

Example

```
$360 ($36) nmdat >wl io_getnext_mib

$b.82020250

$361 ($36) nmdat >io_mib(b.82020250)

RECORD
   MIB_NEXT_REGION              : 8202c050
   MIB_MASTER_LINK              : 82020250
   MIB_NUM_REGIONS              : 0
   MIB_STATE                    : b7
   MIB_FILL_BYTE                : 0
   MIB_MM_EXT_READ_DISABLE_CNT : 0
 ....              ....          ...
 ....              ....          ...
            PI_BLACK_HOLE_PAGE : FALSE
            PI_READ_POSTPONED  : FALSE
END

$364 ($36) nmdat >io_mib(b.82020250,,'mib_master_link')

82020250
```

io_path_to_ldev

Returns the ldev number for the specified path name.

Syntax

```
ldev:int = io_path_to_ldev('n/n.n.n')
```

Parameters

ldev
The logical device number of the device that corresponds to the path specification. -1 is returned if no such ldev exists.

path_spec
The path specification for which a logical device number is desired. (required parameter)

User Discussion

Search the LPT to find the desired path. This will have to be sequential by looking at each entry, extracting the dm_port, searching the PPT for that port # and matching the path of the PPT entry to the path specified.

Example

```
$366 ($36) nmdat > wl io_path_to_ldev('2/4.2.3')

$7
```

io_port_data

Prints the port data area associated with the specified LLIO port.

Syntax

```
return_val:any = io_port_data (portid:int,
                               [detail:int=1],
                               [field_name:str=''])
```

Paramaters

return_val
If the field_name is specified, the value of the field is returned here.

portid
The portid of the LLIO port to print the data area for. (required parameter)

detail
Specify 0 to supress printing; 1 - 5 to cause the specified field (or all fields) to print. The default is 1. (optional parameter)

field_name
Specify the field name to print (depending on the detail) and return. Specify 'pointer' to return the virtual address of the table entry. The default is print all fields in the table entry and return nothing. (optional parameter)

User Discussion

This macro will print fields in the io services port data area only!

Macros Used

port_data

Example

```
$429 ($0) nmdat >io_port_data (fffffa8)

RECORD
   IOS_SEMA                : ffffffff
   IOS_PDA_SIZE            : 140
   MGR_PDA_SIZE            : 408
   MGR_PDA_PTR             : b.80c38340
   MGR_ENTRY               : UNKNOWN
   PORT_NUM                : fffffa8
   AUX_DATA_PTR            : a.c0610000
   AUX_DATA_LEN            : 1ca0
   AUX_RESIDENT            : TRUE
   AUX_FROZEN              : FALSE
   NO_SEND_LOG_TRAPS       : TRUE
   NO_RECV_LOG_TRAPS       : TRUE
   REQUESTORS_REPLY_SUBQ   : 0
    ....                   ...
    ....                   ...
        DESCR_CMPR : IOEQ
        KEY_CMPR   : IOEQ
        MATCH_KEY  : 0
     VP_AUX_DATA   : 0.0
     PORT_NUM      : 0
END
```

io_portid

Returns the portid associated with the specified LLIO path.

Syntax

```
portid = io_portid (path:str)
```

Parameters

portid
The LLIO manager portid corresponding to the specified LLIO path is returned here.

path
Specify the physical device path identification to return the portid for. (required parameter)

User Discussion

Macros Used

```
io_ppt_path
io_ppt
```

Example

```
$2df ($3) nmdat > wl io_portid('8.2.3')

$ffffffc8
```

io_request_table

Stores entries for IO requests made.

Syntax

```
io_request_table
                ([rid_num : INT / U16 = $0] ,
                [detail   : INT / U16 = $1] )
```

Parameters

rid_num
The entry number the user wishes to print. If 0 or no entry number is passed all entries are printed.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Nothing will be printed.
1	Will print the Request ID # and Status of those entries which contain an error status. This is the default.
2	Will format all valid entries. The format will be Request ID #, a Reading flag (True if the action was a read), a CMIO Request flag (True if the request was initiated by Compatibility mode), the State of the request, the IO Function, the LDEV # in decimal, the PIN number, the Status, and the Priority of the request.
> 2	Will execute an 'FV' of the entry.

User Discussion

The IO request table will hold entries for all IO requests made, except Memory Manager Disc I/O and DTC Terminal writes. The free entries are managed with a last in first out algorithm, therefore entries may be few and are not in chronological order.

Example

```
$279 (0) nmdat > io_request_table

                    I/O Request Entries With an Error Status

RID #       STATUS
-----      ---------
No entries to be printed!


$287 (0) nmdat > io_request_table(,2)

                       Formatted I/O Request Table Entries

          R C
          E M
          A I
          D O
          I
          N R
RID #     G Q   STATE                FUNC    LDEV    PIN     STATUS      PRI
-----     - -   ----------------     ----    ----    -----   ---------   -----

$1              IOS_COMPLETED        $4      20      $26     $0          $bb8
$2        T     IOS_COMPLETED        $1      20      $26     $0          $bb8
$3        T     IOS_COMPLETED        $0      177     $32     $0          $bb8
$4        T     IOS_COMPLETED        $1      20      $26     $0          $ab8
```

io_timer_list

Prints the specified number of pending timer requests.

Syntax

```
return_val:any = io_timer_list ([num_requests : int = 0]
                                [detail:int=1])
```

Parameters

num_requests
The number of pending timer requests to print out. (optional parameter)

detail
Specify one of the following to select the level of detail to be printed:

0	Do not print anything; just return the value.
1 - 3	Will print summary information including the state, entry key, expiration time, and reply port.
4 - 5	Will print all L1H-3 information and the message.

The default is 0. (optional parameter)

User Discussion

Example

```
$369 ($36) nmdat > io_timer_list

TIMER          ENTRY   EXPIR.       REPLY
STATE          KEY     TIME         PORT
-----          ---     ----         ----
ACTIVE         $f      $ac672d4d    $ffffff44
ACTIVE         $0      $b870acc8    $ffffffd4
ACTIVE         $10     $dbe0bfad    $ffffff0d
```

io_waiting

Prints the logical input/output operations which have been started and are not complete and are waiting for physical input/output.

Syntax

```
io_waiting ([detail:int=5])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
0 - 5	Print all information about the request.

The default is 5. (optional parameter)

User Discussion

If the summary option is specified, the number of logical input/output operations which have been started and not complete are printed. If the detail option is specified, the process id, ldev, type of i/o (read, write, etc.), unique i/o identifier, filename, file number, and req_address of data are printed for each i/o operation waiting.

Example

Contents

Chapter 7
Process Management Macros

7

Process Management Macros

The Process Management (PM) function includes the following MPE XL functions:

Process Management
Dispatcher
Interrupt Control Stack
Loader

kso__number

Syntax

to be supplied

Parameters

to be supplied

Example

```
$377 ($36) nmdat >wl kso_number('kso_hlio_lpt')
$6E
```

kso_pointer

Syntax

to be supplied

Parameters

to be supplied

Example

```
$378 ($36) nmdat >wl kso_pointer(6e)

$a.c736dbc0
```

pm_devices

Prints all devices owned by the specified process.

Syntax

```
[ldev_list:str] = pm_devices ([pin_num:int=pin],
                              [detail:int=1],
                              [error_parm:str='pad'])
```

Parameters

ldev_list
A string containing a list of the logical device numbers allocated to this process is returned here. This can be used in conjunction with the DAT FOREACH command. (optional parameter)

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print list of ldevs.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

This macro is intended to return and/or print a list of the ldevs "owned" by the specified pin. Device allocation is handled with compatibility mode code. In order to determine which device is allocated, the CM Logical Device Table (LDT) is searched. This table has one entry per logical device and contains the pin of the JSMAIN process associated with the process/job allocating the device. This table needs to be scanned sequentially looking for the JSMAIN pin associated with the specified process.

Macros Used

Example

```
$2ef ($36) nmdat >pm_devices
LDEVS owned by PIN $36: NONE.
```

pm_disp_global

Prints the specified (or all) information in the dispatcher global area.

Syntax

```
[return_val:any] = pm_disp_global ([detail:int=1],
                                   [field_name:str=''])
```

Parameters

return_val

The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. (optional parameter)

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing
1 - 5 Format the specified field in (or the entire) table entry.
The default is 1. (optional parameter)

field_name

Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned. 'pointer' may be specified in the field_name to cause the pointer to the dispatcher global area to be returned. The default is all field names (the entire table). (optional parameter)

User Discussion

The dispatcher gloabal area is used to store information relivant to process management on a system wide scope. This macro is designed to give the user easy access to such info as queue priority limits, queue pointers, and processor status.

Macros Used

none

Example

```
$2e7 ($36) nmdat > pm_disp_global

RECORD
   DGL_ICS_PTR                : 8118a000
   DGL_DISP_EVENT_SET         : [ ]
   DGL_FILLER1                : FALSE
       ...                       ...
       ...                       ...
   DGL_NUM_SWAPINS            : 0
   DGL_SWAPIN_LIMIT           : FALSE
   DGL_SLOW_EVENTS            : [ ]
END


$2e8 ($36) nmdat > pm_disp_global(5)

 READY    ACTIVE    ACTIVE    PENDING     PENDING
 COUNT     PIN       PRI       PIN         PRI
 -----    ------    ------    -------     -------
$0        $36       $bb2       $7ffd      $0

     ES             DS             CS             CS             CS
  QUANTUM        QUANTUM        QUANTUM      QUANTUM MAX    QUANTUM MIN
 ---------      ---------      -------      -----------    -----------
$1a39de0       $1a39de0       $1a39de0     $1a39de0       $68e778


 ES PRIORITY       DS PRIORITY       CS PRIORITY
 BASE    LIM       BASE    LIM       BASE    LIM
 ----    ---       ----    ---       ----    ---
$3e8     $1        $7d0    $3e9      $bb8    $7d1
```

pm_disp_queue

Prints the dispatcher dispatch queue.

Syntax

```
[return_val:str] = pm_disp_queue ([detail:int=1],
                                  [field_name:str=''])
```

Parameters

return_val

The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. The return values are filled in this string such that they can be processed with the DAT FOREACH function. (optional parameter)

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing
1 - 3	Format the specified field or, if no field given, print pin and priority for each element in the queue.
4 - 5	Format the specified field or, if no field given, print the entire pib dispatcher record for each element in the queue.

The default is 1. (optional parameter)

field_name

Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned. The default is all field names (the entire table). (optional parameter)

User Discussion

This macro is used to print information on about all processes that are in the dispatcher queue. At low levels the PIN and priority of each process is printed while at high levels of detail the entire dispatcher section of the pib can be printed. The user also has the ability to extract infomation on specific fields in the dispatcher area of the pib.

Macros Used

pm_disp_global

Example

```
$2ea ($36) nmdat > pm_disp_queue

DISPATCH QUEUE INFO:      PIN      PRIORITY
                          ---      --------
              HEAD ===>   $36        $bb2


$2eb ($36) nmdat > pm_disp_queue(5)

RECORD
   PIB_DISP_LINK                 :
      FWD_LINK : c74003c8
      BWD_LINK : c74003c8
   PIB_PROCESS_INTERRUPT         : [ ]
   PIB_DISP_FILLER2              : FALSE
   PIB_PIN                       : 36
       ....                        .
       ....                        .
   PIB_FILLER4                   : 0
   PIB_PID_FOUR                  : 0
END
```

pm_errors

Prints the error stack associated with the specified process.

Syntax

```
pm_errors ([pin_num:int=pin],
           [detail:int=1],
           [error_parm:str='pad'])
```

Parameters

pin_num
Process id. If omitted, the current process process id number will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing
1 - 3 Print only last five errors for the process.
4 - 5 Print all errors for the process.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

The error records stored for each process contains up to four fields and, up to 64 records are part of a process's pib so printing out all the fields can be time consuming.

Macros Used

Example

```
$2ed ($36) nmdat > pm_errors

ERROR STACK (Entries displayed in most recent to least recent order):

SUBSYSTEM      SUBSYSTEM         SUBSYSTEM
NUMBER         ERROR NUMBER      NAME
---------      ------------      ---------
$6d            $c                Trap Handler
```

pm_family

Prints specified processes immediate family.

Syntax

```
[pin_list:str=''] = pm_family ([pin_num:int=pin],
                               [detail:int=1],
                               [error_parm:str='pad'])
```

Parameters

pin_list
Returns a list of Process Id Numbers (PIN); the first item in the list is the pin of the parent process, the second pin is that of the JSMAIN pin (if any), the third pin is the that of the child pin (if any) and the fourth is the sibling pin (if any). Zero will denote empty entry.

pin_num
process id. If omitted, the current process process id will be used (or ... parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing
1 - 5 Print parent, jsmain, child and sibling.

The default is 1. (optional parameter)

err019c333R or_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

The parent pin, jsmain process (if any), program file and sibling pin are all printed. The returned pin list is in a fixed order, see pin_list above.

Macros Used

pib_info

Example

```
$2f0 ($36) nmdat > pm_family(5)

PARENT PIN  PROGRAM FILE            JSMAIN PIN  PROGRAM FILE
----------  ------------            ----------  ------------
$1          PROGEN.PUB.SYS          NONE        NONE

CHILD PIN   PROGRAM FILE            SIBLING PIN PROGRAM FILE
---------   ------------            ----------- ------------
NONE        NONE                    $6          ..
```

pm_fpib

Prints pib information for a number of processes.

Syntax

```
[all_pins:str] = pm_fpib ([pin_list:str='',
                          [detail:int=7],
                          [error_parm:str='pad'])
```

Parameters

all_pins
A list of all the pins active on the system, generated if the pin_list parameter is null (default).

pin_list
A list of pins to be processed via the FOREACH command (optional parameter)

If an empty list is passed in (default), info for all active pins will be displayed.

detail
Specify a number between 0 and 7 which determines the level of detail to print as specified in the pm_pib_info macro. Detail level 7 prints all pib information (optional parameter).

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

This will basically call pm_pib_info for any number of valid pins, printing the header for the specified detail level on the first pin. If info for all valid pins is desired, pass an empty pin list in (default). Doing this will return a list of all pins, so if one desires to again call fpib (or fpibx), one can pass this pin list in to avoid waiting while the list is generated.

Example

```
$2f2 ($36) nmdat >pm_fpib
Formatted Process Information Blocks:


(part 1)  ======= FAMILY TREE =======          =====  MISC VALUES  =====
                                                 CM                      SWITCH  SWITCH
                                                 TOS      JSMAIN          TO      TO
PIN  STATE  PARENT   JSMAIN   CHILD   SIBLING  LEN      PORT            CM      NM
---  -----  ------   ------   -----   -------  ---      ------          ------  -----
$1   ALIVE  $0       $0       $2      $0       $e50     $0              $0      $0
$2   ALIVE  $1       $0       $0      $3       $17f0    $0              $0      $0
$3   ALIVE  $1       $0       $0      $4       $e50     $0              $0      $0
 .    ..     .        .        .       .        ..       .               .       .
 .    ..     .        .        .       .        ..       .               .       .
$36  ALIVE  $17      $17      $0      $0       $f50     $0              $1      $0

(part 2)  ===========  PORT INFO  ===========  =====  INTERRUPT INFO  =====

      P   P P     A P
      O   R R H L P P
      R   I I I O R R                                          PROCESS
      T P A D I I I I              NAME or NUM                 INTRPT  DISP DELAYED

      I F D E N I N N PRI   SEM    WAITED                      DISABLE INT  INT
PIN   N P J L T T T T BST   RANK   PORT                        COUNT   LEVL LEVL
---   - - - - - - - - ---   ----   ----------------            -----   ---- ----
$1    T F T F F F T F $5    $0     PROGEN_GLOBAL_PORT          $0      $6   $6
$2    T F F F F F T F $0    $0     STD SIGNAL PORT             $0      $6   $6
$28   T F F F F F F F $0    $0     JSMAIN_PORT                 $0      $6   $6
  .     ....          .     .          ......                 .       .    .
  .     ....          .     .          ......                 .       .    .
$36   T F F F F F F F $2    $0     STD MESSAGE PORT            $0      $6   $6

(part 3)  ===========  STANDARD PORT INFO ===========

          SIGNAL PORT (wait events)
          -----------
      N G L M       J T     F
      M R R A B   R U I M S T I S D B
      I I I I I I I N M S O H M I E R                          INTERRUPT PORT
PIN   O N N L O O T K E G N R P R L K MESSAGE PORT             (nonempty subq)
---   - - - - - - - - - - - - - - - - ------------            -------------
$1
$2                  J
$3                  J
$4
$5
$6
$7
$8
$9
```

```
$a
$b                    M
$c            I       M        D
$d                    M
$e                    M
$f                    M
$10
$11
$12                                System Logging Wait
$13
 .
 .
$35
$36                                Wait for fill disc request

(part 4)  ============  DISPATCH INFO  ============

      S I   D   T
      W N B B F R S
      A T P P S A Y
      P P E E W N S
      A N N N A C P
      T D B B P M R                                    PEND
      T N L L I P O   SCHED         SCHED              EVENT  WAIT    LAST       HOLD
PIN   N G D D N L C   STATE         CLASS    PRI       SET    EVENT   PRI         PRI
---   - - - - - - -   -----         -----    ---       -----  -----   ----        ---
$1    F F F F F T     LONG_WAIT     AS       $7918            IPC     $0
$2    F F F F F F     LONG_WAIT     BS       $38ff            IPC     $0
 .      ...             ...          .        ..               ..      .
 .      ...             ...          .        ..               ..      .
$35   F F F F F F     LONG_WAIT     ES       $3e8             IPC     $0
$36   F F F F F F     EXECUTING     CS       $bb2             NONE    $0
```

```
(part 5) ============= NM STACK PTR INFO ============

PIN   BASE          LIMIT         MAXDATA     SP-INIT     SP-MAX        PCBX
---   ----          -----         -------     -------     ------        ----
$1    $b.40200000   $b.40230000   $0.0        $0.0        $0.0          $40011db0
$2    $29.40200000  $29.40260000  $0.0        $0.0        $29.40260000  $40010db0
 .       ....          ....         .           .           ...           ...
 .       ....          ....         .           .           ...           ...
$35   $111.402016c0$111.402616c0$0.0          $0.0        $111.402616c0$40010db0
$36   $11c.40200400$11c.40260400$0.0          $0.0        $11c.40260400$40010cb0

(part 6)  ==============  MISCELLANEOUS INFO  ============

          B B   B
      I   R R   R
      N S K K   K
      I T R R   R
      N K D C   P
      T L O N   E    CM
      P C N C   N    TOS            HEAP          PROTCT       SPACE
PIN   P K E L   D    START          PTR           ID           ID
---   - - - -   -    -----          ----          ------       -----
$1    F T F $0E $0   $b.400211b0    $0.0          $3           $b
$2    F T F $0E $0   $29.4001f810   $0.0          $f           $29
 .      ...  .   .      ....          .            .            .
 .      ...  .   .      ....          .            .            .
$35   F T F $0E $0   $111.400201b0  $0.0          $80          $111
$36   F T F $0E $0   $11c.400200b0  $0.0          $85          $11c

$2f6 ($36) nmdat >pm_fpib(,1)

Formatted Process Information Blocks:

(part 1)  ======= FAMILY TREE =======        =====  MISC VALUES  =====
                                                 CM                    SWITCH  SWITCH
                                                 TOS      JSMAIN       TO      TO
PIN   STATE   PARENT   JSMAIN   CHILD   SIBLING  LEN      PORT         CM      NM
---   -----   ------   ------   -----   -------  ---      ------       ------  -----
$1    ALIVE   $0       $0       $2      $0       $e50     $0           $0      $0
$2    ALIVE   $1       $0       $0      $3       $17f0    $0           $0      $0
 .     ..      .        .        .       .        ..       .            .       .
 .     ..      .        .        .       .        ..       .            .       .
$35   ALIVE   $18      $0       $0      $0       $e50     $0           $0      $0
$36   ALIVE   $17      $17      $0      $0       $f50     $0           $1      $0
```

pm_fpibx

Prints pibx information for a number of processes.

Syntax

```
[all_pins:str] = pm_fpibx ([pin_list:str='',
                           [detail:int=5],
                           [error_parm:str='pad'])
```

Parameters

all_pins
A list of all the pins active on the system, generated if the pin_list parameter is null (default).

pin_list
A list of pins to be processed via the FOREACH command (optional parameter)

If an empty list is passed in (default), info for all active pins will be displayed.

detail
Specify a number between 0 and 5 which determines the level of de print as specified in the pm_pibx_info macro. Detail level 5 (the default) prints all pibx information (optional parameter).

error_purm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

This will basically call pm_pibx_info for any number of valid pins, printing the header for the specified detail level on the first pin. If info for all valid pins is desired, pass an empty pin list in (default). Doing this will return a list of all pins, so if one desires to again call fpibx (or fpib), one can pass this pin list in to avoid waiting while the list is generated.

Example

```
$2f7 ($36) nmdat >pm_fpibx(,1)

Formatted Process Information Block eXtensions:


(part 1) ====== VARIOUS BOOLEANS ======  ====== PIDS/PINS TO REMEMBER ======

    A     I T
    P   C S R U
    A D M T A N
    R E M R P L
    E B O I B O
    N U D N I A                          INIT   RECENT   WAITED   WAITED   XDB
PIN T G E G T D RFILES                   PID    PID      PID      KILLER   TOOLSET PIN
--- - - - - - - ------                   ----   ------   -------  ------   ---------
-
$1  F F F F F F (F F F F F F F F) $0      $1     $0       $0       $0
$2  F F T F F T (F F F F F F F F) $0      $0     $0       $0       $0
$3  T F F F F F (F F F F F F F F) $0      $0     $0       $0       $0
 .    ...           ...           .      .      .        .        .
 .    ...           ...           .      .      .        .        .
$35 F F F T F T (F F F F F F F F) $1      $0     $0       $0       $0
$36 F F F F F T (F F F F F F F F) $1      $5     $0       $0       $0

$2f8 ($36) nmdat >pm_fpibx(,5)

Formatted Process Information Block eXtensions:


(part 1) ====== VARIOUS BOOLEANS ======  ====== PIDS/PINS TO REMEMBER ======

    A     I T
    P   C S R U
    A D M T A N
    R E M R P L
    E B O I B O
    N U D N I A                          INIT   RECENT   WAITED   WAITED   XDB
PIN T G E G T D RFILES                   PID    PID      PID      KILLER   TOOLSET PIN
--- - - - - - - ------                   ----   ------   -------  ------   ---------
-
$1  F F F F F F (F F F F F F F F) $0      $1     $0       $0       $0
$2  F F T F F T (F F F F F F F F) $0      $0     $0       $0       $0
 .     ....         ....          .      .      .        .        .
 .     ....         ....          .      .      .        .        .
$35 F F F T F T (F F F F F F F F) $1      $0     $0       $0       $0
$36 F F F F F T (F F F F F F F F) $1      $5     $0       $0       $0
```

```
(part 2) ==================  POINTER INFO  ====================

                DEP     CCB          FEQ        NOTIF      PID
PIN CCB         LIST    PTR          TAB        LIST       LIST      UIT HDR
--- ---         ----    -----        ---        -----      ----      -------
$1  $0          $0      $0           $0         $c04c0064  $0        $0
$2  $0          $0      $0           $0         $0         $0        $0
 .      ...      .         ...        .          ...       .          ...
 .      ...      .         ...        .          ...       .          ...
$35 $0          $0      $0           $0         $c04c003c  $0        $0
$36 $c2c38000   $0      $c2c38000    $0         $0         $0        $c2c08000

(part 3) ====================  LOADER INFO  ====================

     CM                 CM                                                LDR
     AREA               AREA               ENTRY      ENTRY   ENTRY      SYMBOL
PIN  BASE               LIMIT              DP         OFFSET  SID        PTR
---  ----               -----              -----      ------  -----      ------
$1   $b.40001000        $b.40032000        $0         $0      $0         $0.0
$2   $29.40000000       $29.40031000       $0         $0      $0         $0.0
 .       ....               ....           .          .       .          ..
 .       ....               ....           .          .       .          ..
$35  $111.40000000      $111.40031000      $40200008  $b7dc   $114       $0.0
$36  $11c.40000000      $11c.40031000      $40200008  $938c   $70        $0.0

(part 4) ====================  LOADER INFO cont.  ===============
    HEAP          HEAP      HEAP          HEAP      STACK                    XRT
PIN BASE          DIR       LIMIT         SIZE      SIZE      XRT            SIZE
--- ----          ----      -----         ----      -----     ---            ----
$1  $0.0          $0        $0.0          $0        $30000    $0.0           $0
$2  $29.40370000  $0        $29.46518000  $61a8000  $60000    $29.40100000   $100000
 .      ....       .           ....          ...       ...       ....           ...
 .      ....       .           ....          ...       ...       ....           ...
$35 $111.40372000 $0        $111.4651a000 $61a8000  $60000    $111.40100000  $100000
$36 $11c.40371000 $0        $11c.46519000 $61a8000  $60000    $11c.40100000  $100000
```

pm_ics_header

Prints the specified (or all) information in the ICS header area.

Syntax

```
[return_val:str] = pm_ics_header ([detail:int=1],
                                  [field_name:str=''])
```

Parameters

return_val

The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. (optional parameter)

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing

1 - 5 Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name

Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned. 'pointer' may be specified in the field_name to cause the pointer to the ics header area to be returned. The default is all field names (the entire table). (optional parameter)

User Discussion

This macro is used to return global information about the ICS.

Macros Used

```
kso_number
kso_pointer
```

Example

```
$2f9 ($36) nmdat > pm_ics_header

RECORD
   ICS_INT_NEST_COUNT : 0
   ICS_SIZE           : 28000
      ...                ...
      ...                ...
         SR5         : 6007b
         ESCC        : 4007b
         ISM_FILLER  : 5007b
END
```

```
$2fd ($36) nmdat > pm_ics_header(5,'ics_size')

28000
```

pm_job

Prints the job number and logon information associated with the specified process.

Syntax

```
[job_num:int] = pm_job ([pin_num:int=pin],
                        [detail:int=1],
                        [error_parm:str='pad'])
```

Parameters

job_num
The job/session number associated with the specified process is returned here. A negative number indicates a batch job; a positive number indicates a session. (optional parameter) ??? check to make sure this fits with the convention.

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing
1 - 5	Print job number and user name.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Prints the job/session number and the logon information associated with the specified pin.

Macros Used

pibx_info

Example

```
$2fe ($36) nmdat > pm_job
JOB NUMBER: #S1
```

pm__kpo__number

Returns the number of the specified Known Process Object constant.

Syntax

```
kpo_num:int = pm_kpo_number (kpo_name:str)
```

Parameters

kpo__num
The Known Process Object number corresponding to the specified kpo__name. (required parameter)

kpo__name
The name to look up the KPO number for. (required parameter)

User Discussion

The number of the specified KPO is returned. An error is generated of the KPO does not exist.

Macros Used

none

Example

```
$361 ($36) nmdat > wl pm_kpo_number('kpo_file_system')

$4e31
```

pm_kpo_pointer

Returns the virtual address of the specified KPO for the specified process.

Syntax

```
kpo_address:ptr = pm_kpo_pointer
                  ([pin_num:int=pin],
                  pm_kpo_number:int)
```

Parameters

kpo_address
The virtual address of the start of the object corresponding to the specified KPO number is returned here. (required parameter)

pin_num
The process id to return the KPO for. If not specified, the current process is assumed. (optional parameter)

pm_kpo_number
The number of the KPO to return the address for. This can be obtained using the pm_kpo_number macro. (required parameter)

User Discussion

The virtual address of the specified KPO is returned. If the KPO does not exist or is not a valid KPO, an error is generated.

Macros Used

none

Example

```
$365 ($36) nmdat > wl pm_kpo_pointer(,pm_kpo_number('kpo_file_system'))

$a.c2c216e0
```

pm_loaded_file_table

Prints the specified (or all) information in the Loaded File Table for the specified UFID.

Syntax

```
[return_val:str] = pm_loaded_file_table ([ufid:str=''],
                                         [detail:int=1],
                                         [field_name:str=''])
```

Parameters

return_val
The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. (optional parameter)

ufid
The Unique File ID of the file requested. The this can be obtained using the file system macros. If not specified, the entire table will be printed. (optional parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned. The default is all field names (the entire table). Specify 'pointer' to have the address of the table entry returned. (optional parameter)

User Discussion

Macro not emplemented at first release.

Macros Used

kso_number kso_pointer

Example

pm_loader_globals

Prints the specified (or all) information in the Loader System Globals area.

Syntax

```
[return_val:str] = pm_loader_globals ([detail:int=1],
                                      [field_name:str=''])
```

Parameters

return_val
The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. (optional parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned.

'pointer' may be specified in the field_name to cause the pointer to the loader globals area to be returned.

The default is all field names (the entire table). (optional parameter)

User Discussion

Primarily to be used to return values from the loader globals area.

Macros Used

```
kso_number
kso_pointer
```

Example

pm_pib_info

Prints all information contained in the PIB. Optionally returns the value of a selected field.

Syntax

```
[return_value:any] = pm_pib_info ([pin_num:int=pin],
                                  [next_pin:boolean=false],
                                  [detail=7],
                                  [field_name=''],
                                  [header = TRUE])
```

Parameters

return_value
If the field_name parameter is specified, the value of that field is returned here (optional parameter).

pin_num
The process identification number (pin) of the PIB to print. If omitted, the current process will print (optional parameter).

next_pin
Specify true to return the number of the next active pin in the PIB. If there is no next pin, -1 is returned.
The default is false. (optional parameter)

detail
Specifies the amount of detail to be printed. Select one of the following (each level includes the level preceeding it):

0	Do not print anything.
1	Print family tree and miscellaneous information.
2	Print port area and interrupt information.
3	Print process standard port information.
4	Print dispatch information.
5	Print stack pointer information.
6	Print miscellaneous booleans and pointers.
> 6	Print all information.

field_name
To return the value of a field, specify the field name as defined in the PIB include file.

header
If true than a header will be printed with the information. (optional parameter)

User Discussion

The user can either choose to print ha header or not to print a header with the information. This is accomplished by setting the header parameter. If the field_name parameter is used, the header is ignored and the value of the field is either returned or printed depending on the value of the detail parameter.

Example

```
$2f4 ($36) nmdat > pm_pib_info

(part 1)  ======= FAMILY TREE =======        =====  MISC VALUES  =====
                                                CM                    SWITCH  SWITCH
                                                TOS     JSMAIN        TO      TO
PIN  STATE  PARENT   JSMAIN   CHILD    SIBLING LEN     PORT          CM      NM
---  -----  ------   ------   -----    ------- ---     ------        ------  -----
$36  ALIVE  $17      $17      $0       $0      $f50    $0            $1      $0

(part 2)  ===========  PORT INFO  ===========  =====  INTERRUPT INFO  =====

      P   P P     A P
      O   R R H L P P
      R   I I I O R R                                          PROCESS
      T P A D I I I I             NAME or NUM                  INTRPT  DISP DELAYED
      I F D E N I N N PRI  SEM    WAITED                       DISABLE INT  INT
PIN   N P J L T T T T BST  RANK   PORT                         COUNT   LEVL LEVL
---   - - - - - - - - ---  ----   ----------------           -----   ---- ----
$36   T F F F F F F F $2   $0     STD MESSAGE PORT             $0      $6   $6

(part 3)  ===========  STANDARD PORT INFO ===========

         SIGNAL PORT (wait events)
         -----------
     N G L M       J T     F
     M R R A B   R U I M S T I S D B
     I I I I I I I N M S O H M I E R                          INTERRUPT PORT
PIN  O N N L O O T K E G N R P R L K MESSAGE PORT            (nonempty subq)
---  - - - - - - - - - - - - - - - - ------------            --------------
$36                                  Wait for fill disc request

(part 4)  ============  DISPATCH INFO  ============

      S I   D   T
      W N B B F R S
      A T P P S A Y
      P P E E W N S
      A N N N A C P
      T D B B P M R                                  PEND
      T N L L I P O    SCHED         SCHED          EVENT WAIT    LAST      HOLD
PIN   N G D D N L C    STATE         CLASS  PRI     SET   EVENT   PRI       PRI
---   - - - - - - -    -----         -----  ---     ----- -----   ----      ---
```

```
$36   F F F F F F      EXECUTING     CS     $bb2        NONE    $0

(part 5) ============= NM STACK PTR INFO ============

PIN   BASE          LIMIT         MAXDATA     SP-INIT     SP-MAX       PCBX
---   ----          -----         -------     -------     ------       ----
$36   $11c.40200400$11c.40260400$0.0        $0.0        $11c.40260400$40010cb0

(part 6)  ==============  MISCELLANEOUS INFO  ============

          B B   B
      I   R R   R
      N S K K   K
      I T R R   R
      N K D C   P
      T L O N   E    CM
      P C N C   N    TOS           HEAP          PROTCT       SPACE
PIN   P K E L   D    START         PTR           ID           ID
---   - - - -   -    -----         ----          ------       -----
$36   F T F $0E $0   $11c.400200b0 $0.0         $85          $11c
$2f6 ($36) nmdat > pm_pib_info(,,1)

(part 1)  ======= FAMILY TREE =======        =====  MISC VALUES  =====
                                                CM                SWITCH SWITCH
                                                TOS    JSM        TO     TO
PIN   STATE   PARENT   JSMAIN   CHILD    SIBLING LEN    PO?        CM     NM
---   -----   ------   ------   -----    ------- ---    ---        ------ -----
$36   ALIVE   $17      $17      $0       $0      $f50   $0         $1     $0


$2f7 ($36) nmdat > pm_pib_info(,,2)

(part 2)  ===========  PORT INFO  ===========  =====  INTERRUPT INFO  =====

      P   P P     A P
      O   R R H L P P
      R   I I I O R R                                    PROCESS
      T P A D I I I I            NAME or NUM             INTRPT  DISP DELAYED
      I F D E N I N N PRI  SEM   WAITED                  DISABLE INT  INT
PIN   N P J L T T T T BST  RANK  PORT                    COUNT   LEVL LEVL
---   - - - - - - - - ---  ----  ----------------      -----   ---- ----
$36   T F F F F F F F $2   $0    STD MESSAGE PORT        $0      $6   $6
```

```
$2f8 ($36) nmdat > pm_pib_info(,,3)

(part 3)  ===========  STANDARD PORT INFO ===========

          SIGNAL PORT (wait events)
          -----------
      N G L M       J T     F
      M R R A B   R U I M S T I S D B
      I I I I I I I N M S O H M I E R                              INTERRUPT PORT
PIN   O N N L O O T K E G N R P R L K MESSAGE PORT                 (nonempty subq)
---   - - - - - - - - - - - - - - - - ------------                 --------------
$36                                  Wait for fill disc request
```

```
$2fa ($36) nmdat > pm_pib_info(,,4)

(part 4)  ============  DISPATCH INFO  ============
       S I   D   T
       W N B B F R S
       A T P P S A Y
       P P E E W N S
       A N N N A C P
       T D B B P M R                                  PEND
       T N L L I P O    SCHED          SCHED          EVENT WAIT     LAST       HOLD
PIN    N G D D N L C    STATE          CLASS  PRI     SET   EVENT    PRI        PRI
---    - - - - - - -    -----          -----  ---     ----- -----    ----       ---
$36    F F F F F F      EXECUTING      CS     $bb2           NONE     $0
```

```
$2fb ($36) nmdat > pm_pib_info(,,5)

(part 5) ============= NM STACK PTR INFO ============

PIN    BASE          LIMIT         MAXDATA     SP-INIT     SP-MAX        PCBX
---    ----          -----         -------     -------     ------        ----
$36    $11c.40200400$11c.40260400$0.0         $0.0        $11c.40260400$40010cb0
```

pm_pibx_info

Prints all information contained in the PIB extension. Optionally returns the value of a selected field.

Syntax

```
[return_value:any] = pm_pibx_info ([pin_num:int=pin],
                                   [detail:int=5],
                                   [field_name:str=''],
                                   [header : bool = true])
```

Parameters

return_value
If the field_name parameter is specified, the value of that field is returned here (optional parameter).

pin_num
The process identification number (pin) of the PIBX to print. If omitted, the current process will print (optional parameter).

detail
Specifies the amount of detail to be printed. Select one of the following (each level includes the level preceeding it):

0	Do not print anything.
1	Print various booleans and the critical pids and pins saved in the pibx.
2	Print pointer information.
3	Print the loader information.
4	Print miscellaneous information.
5	Print all of L1H through 4 information.

field_name
To return the value of a field, specify the field name as defined in the PIBX include file.

header
If true then the information form the pibx wil be printed with a header.

User Discussion

The user can either print the information with a header or without a header. If the field_name parameter is specified, header is ignored and the value of the field_name is either returned or printed depending on the value of the detail parameter.

Macros Used

Example

pm_process_file_list

Prints the specified (or all) information in the Process File List (PFL).

Syntax

```
[return_val:any] = pm_process_file_list ([pin_num:int=pin],
                                         [entry:int=0],
                                         [detail:int=1],
                                         [header_info:bool=false],
                                         [field_name:str=''])
```

Parameters

return_val
The value of the specified field_name is returned here. If the field_name is not specified, nothing is returned. This only applies to information in the pfl table header or if a specific entry is requested (see entry). (optional parameter)

pin_num
Specify the process id number of the process to print the PFL for. If not specified, the current process will be used. (optional parameter)

entry
Specify the relative entry number of the PFL entry to print. If not specified, all entries will print. (optional parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing
1 - 5 Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

header_info
If TRUE return info on the pfl_header otherwise, return field is from a pfl_entry. Default is False. (optional parameter)

field_name
Specify the field name to select a field to be printed or returned. Only the selected field name will be printed or returned. The default is all field names (the entire record). (optional parameter)

User Discussion

This macro is intended to be used to return information is a processes PFL table. The PFL is a KPO. use the format_table macro to format the table.

Macros Used

Example

```
$301 ($36) nmdat > pm_process_file_list

RECORD
   NEXT             : c4d0831c
   PREV             : 0
   PFL_HDR          : c4d08250
   GLOBAL_INFO      : 850241b0
   NAME             : 'NL.PUB.SYS'
   FID              : 0
   NUM_LOADED_SOMS  : 1
   SOM_DATA         : c4d08308
   LAST_USER_FILE   : FALSE
   MEMRES           : FALSE
END

RECORD
   NEXT             : 0
   PREV             : c4d082bc
   PFL_HDR          : c4d08250
   GLOBAL_INFO      : 85025a70
   NAME             : 'CI.PUB.SYS'
   FID              : 8
   NUM_LOADED_SOMS  : 1
   SOM_DATA         : c4d08368
   LAST_USER_FILE   : TRUE
   MEMRES           : FALSE
END


$303 ($36) nmdat > pm_process_file_list(,,5,,'name')

'NL.PUB.SYS'


'CI.PUB.SYS'
```

pm_ptree

Prints a process tree for a specified process.

Syntax

```
[pin_list:str=''] = pm_ptree ([start_pin_num :int =1],
                              [detail        :int =5],
                              [indentation   :int =1])
```

Parameters

pin_list
A list of the pins printed by this macro is returned here such that it can be used with the DAT FOREACH command. (optional parameter)

start_pin_num
The pin at the root of the requested tree. The default is pin 1. (optional parameter)

detail
Specify a number between 1 and 5 which determines the level of d print as follows:

0	Do not print anything, just return the pin_list.
1	Print pins only.
2	Print pins and files being executed.
3	Print pins and job/session numbers.
4 - 5	Print pins, files, and job/session numbers.

The default is 5 (optional parameter).

indentation
For DAT Macro internal usage. Do not specify this parameter. Indicates a desired starting indentation, each level of child pins will be indented four additional columns (optional parameter).

User Discussion

This macro will produce output similar to the DAT DPTREE command which will be removed when the macro is implemented. It can be used interchangably with the DPTREE command.

Macros Used

```
pib_info
pibx_info
format_job
```

Example

```
$303 ($37) nmdat >pm_ptree

$1  (PROGEN.PUB.SYS)
    $2  (LOAD.PUB.SYS)
    $3  (..)
    $4  (..)
    $5  (..)
    $6  (..)
    $7  (..)
    $8  (..)
    $9  (..)
    $a  (..)
    $b  (..)
        $d  (NMFILE.PUB.SYS)
        $e  (NMLOGMON.PUB.SYS)

            $f  (NMLOGMON.PUB.SYS)
    $c  (..)
    $10 (POOMBMGR.PUB.SYS)
    $11 (..)
    $12 (LOG.PUB.SYS)
    $13 (SYSMAIN.PUB.SYS)
    $15 (SESSION.PUB.SYS)
        $17 (JSMAIN.PUB.SYS)
            $36 (CI.PUB.SYS)  #S1
        $19 (JSMAIN.PUB.SYS)
       $1a (JSMAIN.PUB.SYS)
        $1d (JSMAIN.PUB.SYS)
        $14 (JSMAIN.PUB.SYS)
        $20 (JSMAIN.PUB.SYS)
        $22 (JSMAIN.PUB.SYS)
        $24 (JSMAIN.PUB.SYS)
        $26 (JSMAIN.PUB.SYS)
        $28 (JSMAIN.PUB.SYS)
        $29 (JSMAIN.PUB.SYS)
        $2a (JSMAIN.PUB.SYS)
        $2b (JSMAIN.PUB.SYS)
        $2c (JSMAIN.PUB.SYS)
        $2d (JSMAIN.PUB.SYS)
        $2e (JSMAIN.PUB.SYS)
        $2f (JSMAIN.PUB.SYS)
        $30 (JSMAIN.PUB.SYS)
        $31 (JSMAIN.PUB.SYS)
        $32 (JSMAIN.PUB.SYS)
        $33 (JSMAIN.PUB.SYS)
    $16 (JOB.PUB.SYS)
        $1b (JSMAIN.PUB.SYS)
        $1c (JSMAIN.PUB.SYS)
        $1e (JSMAIN.PUB.SYS)
        $1f (JSMAIN.PUB.SYS)
        $21 (JSMAIN.PUB.SYS)
```

```
          $23 (JSMAIN.PUB.SYS)
          $25 (JSMAIN.PUB.SYS)
          $27 (JSMAIN.PUB.SYS)
      $18 (DIAGMON.DIAG.SYS)
          $34 (MEMLOGP.DIAG.SYS)
          $35 (LOGGER.DIAG.SYS)
$1 nmdat> pm_ptree (7,1)

7
    9
        d
    e

$1 nmdat> pm_ptree (7,4)

7   (SESSION.PUB.SYS)
    9   (JSMAIN.PUB.SYS) #S1
        d   (CI.PUB.SYS) #S1
    e   (JSMAIN.PUB.SYS) #S2
```

pm_scheduling

Prints the scheduling information associated with the specified process.

Syntax

```
pm_scheduling ([pin_num:int=pin],
               [detail:int=1],
               [error_parm:str='pad'])
```

Parameters

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing.

1 - 5 Print out the scheduling info for the specified pin.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Prints information about the specified pins current scheduling state.

Macros Used

```
clock_time
```

Example

```
$308 ($36) nmdat >pm_scheduling(1)

        PROCESS
QUEUE   STATE       PRIORITY   CPU TIME     REAL TIME     CUMULATIVE CPU - REAL
-----   -----       --------   --------     ---------     --------------   ----
AS      LONG_WAIT   $7918      $8b077c52                  $0
```

pm_semaphores

Prints the semaphore waited on by a specified process (if any).

Syntax

```
[address:ptr=NIL] = pm_semaphores ([pin_num:int=pin],
                                   [detail:int=1],
                                   [error_parm:str='pad'])
```

Parameters

address
The address of the semaphore being waited on is returned here. If the process is not waiting on a semaphore, NIL is returned. (optional parameter)

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print semaphore address.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Since the management of semaphores is such a low-level and performance sensitive function, the operating system does not track which semaphores are owned by a process. What this macros tells us is what we are waiting on not what we own.

There is a field in the PIB in the semaphore section (presumably) which contains the address of the semaphore being waited on. Print/return this address if it is not zero.

Macros Used

pib_info

Example

```
$30b ($36) nmdat >pm_semaphores(1)

NOT WAITING ON A SEMAPHORE.
```

pm_trace

Prints a stack marker trace for the specified process.

Syntax

```
pm_trace ([pin_num:int=pin],
                [detail:int=1],
                [error_parm:str='pad'])
```

Parameters

pin_num
process id. If omitted, the current process process id will be used (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1	tr single.
2	tr dual.
3	tr unwind.
4	tr ism.
5	tr full.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Various stack traces are printed using the standard DAT command "tr". See DAT manual for discussion of the tr command.

Macros Used

none

Example

```
$2eb ($36) nmdat > pm_trace

...unwinding out of RDB:
R1=$1f0000b4 := $c01066e0
R2=$418 := $293a9c
R3=$4 := $400114a0
R4=$9 := $c2c20000
R5=$30000000 := $101000
R6=$40011484 := $40011484
R7=$11c := $11c
R8=$40011432 := $40011432
R9=$10 := $10
R10=$13 := $13
R11=$fffffffe0 := $fffffffe0
R12=$11c := $11c
R13=$11c := $11c
R14=$10 := $10
R15=$10 := $10
R16=$fff0 := $fff0
R17=$40201f9c := $40201f9c

R18=$3 := $3
R19=$94598 := $4020368c
R20=$20000 := $a
R21=$20020 := $293a94
R22=$c01061a0 := $c01061a0
R23=$11c := $11c
R24=$20 := $79
R25=$3 := $c006d
R26=$1 := $fc190064
R27=$c0200008 := $c0200008
R28=$c006d := $c006d
R29=$cf000 := $cf000
R30=$40202190 := $40202190
R31=$0 := $0
RCTR=$0 := $0
PID1=$10a := $10a
PID2=$64 := $64
CCR=$0 := $0
SAR=$1d := $18
PID3=$0 := $0
PID4=$0 := $0
```

```
IVA=$8c000 := $8c000
EIEM=$30000000 := $ffffffff
ITMR=$424da657 := $35c8102d
PCSF=$0 := $a
PCOF=$96a64 := $2975bc
IIR=$485f0000 := $0
ISR=$a := $a
IOR=$418 := $0
IPSW=$8 := $4f80b
EIRR=$82000000 := $0
TR0=$5e4a00 := $5e4a00
TR1=$624a00 := $624a00
TR2=$0 := $96818
TR3=$40202190 := $0
TR4=$cf000 := $400114a0
TR5=$fff58000 := $c2c20000
TR6=$418 := $101000
TR7=$40011484 := $40011484
SR0=$a := $a
SR1=$a := $a

SR2=$0 := $0
SR3=$0 := $0
SR4=$a := $a
SR5=$11c := $11c
SR6=$b := $b
SR7=$a := $a
PCSB=$0 := $a
PCOB=$0 := $2975c0
PSW=$4f80b := $4f80b
       PC=a.002975bc system_abort
NM* 0) SP=40202190 RP=a.00293a9c ?system_abort+$528
         export stub: a.00455688 nm_switch_code+$b3c
NM  1) SP=40202190 RP=a.000b2e70 SWT_RETURN
       (switch marker frame)
   CM       SYS  %  27.253    SWITCH'TO'NM'+%4       (MitroC CCG)  SUSER1
   CM *  0) SYS  %  27.253    SWITCH'TO'NM'+%4       (MitroC CCG)  SUSER1
   CM    1) SYS  %   7.13342  HPNEWALTXXX+%360       (MitroC CCG)  CISEG4
   CM    2) SYS  %   7.14253  CXNEWGROUP+%506        (MitroC CCG)  CISEG4
   CM    3) SYS  %   6.12303  EXEC_CMD+%615          (MitroC CCG)  CISEG3
   CM    4) SYS  %   6.13456  RECEIVE'CX'SWIT+%56    (Mitroc CCG)  CISEG3
```

```
   CM    5) switch marker                                  (Mitroc CCG)
NM  2) SP=40201dc8 RP=a.000b2bf4 ?CM_SWITCH+$38
         export stub: a.0045453c switch_to_cm+$884
NM  3) SP=40201bd8 RP=a.00719c44 try_exec_cmd.do_cx_switch+$190
NM  4) SP=40201818 RP=a.00719d5c try_exec_cmd+$fc
NM  5) SP=402015a8 RP=a.0071a284 command_interpret+$2f8
NM  6) SP=40201168 RP=a.0071a4ec xeqcommand+$140
NM  7) SP=40200de8 RP=a.0047941c exec_usercmd+$2ec
NM  8) SP=40200d68 RP=a.00717924 exec_cmdfile+$40
NM  9) SP=402009c0 RP=a.00719e44 try_cmdf_or_progf+$b8
NM  a) SP=40200980 RP=a.0071a320 command_interpret+$394
NM  b) SP=40200938 RP=a.0071a66c xeqcommand+$2c0
NM  c) SP=402005b8 RP=a.0071821c ?xeqcommand+$8
         export stub: 70.00011e00
NM  d) SP=40200538 RP=70.000097cc
NM  e) SP=402004f8 RP=70.00000000
     (end of NM stack)
```

Contents

Chapter 8
Resource Management Macros

8

Resource Management Macros

This section is concerned with the various system resources (mainly semaphores) to be examined while looking at a memory dump.

rm_format_sirs

Formats the compatability mode SIR table.

Syntax

```
pin_list:str = rm_format_sirs([detail:int=5])
```

Parameters

pin_list

Returns a list of strings of PIN numbers. Each string of PIN numbers is separated by a space for processing by DAT command FOREACH. In turn, each PIN number is separated from other PIN numbers by a space (optional parameter).

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print the formatted SIR table and indicate whether any deadlocks are found.

The default is 5. (optional parameter)

The compatability mode SIR table is formatted and printed. The output list is made up of lists of PINs that are derived from rm_sir_blocked_proc.

User Discussion

Example

```
$317 ($36) nmdat > rm_format_sirs

************************
SIR Table (Locked SIRs)
************************
No SIRs are locked!
```

rm_global_deadlock

Indicates if any deadlocks or resource contention problems exist between native mode semapaphores and compatability mode sirs.

Syntax

```
pin_list:str = rm_global_deadlock ([detail:int=1])
```

Parameters

pin_list
Return a string list of PINs that are deadlocked (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 5 Print deadlock information.

The default is 1. (optional parameter) Currently this parameter was not used.

This macro will use a deadlock detection algorithm to find resource contention problems that may occur between native mode semaphores and compatability mode sirs.

User Discussion

Example

```
$319 ($36) nmdat > rm_global_deadlock

Getting semaphore info for ALL pins ...
Getting SIR info for ALL pins ...

 No Deadlock detected.
```

rm_sem_blocked_proc

Print (or return) the processes that are blocked (impeded) on the specified semaphore.

Syntax

```
pin_list:str = rm_sem_blocked_proc
               (sem_addr:ptr,
               [detail:int=5])
```

Parameters

pin_list
Returns a list of PINs that are blocked on the specified semaphore. (optional parameter)

sem_addr
The virtual address of the semaphore that information is requested about. Required parameter.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print a list of process that are blocked on the specified semaphore.

The default is 5. (optional parameter)

This macro is designed to print all PINs that are waiting on the specified semaphore. The return string is a list of PINs suitable for processing with the DAT command FOREACH.

User Discussion

Example

```
$31a ($36) nmdat > rm_sem_blocked_proc(1)

List of pins waiting on semaphore at $00000001
```

rm_sem_deadlock

Indicates if any deadlocks are detected for native mode semaphores.

Syntax

```
pin_list:str = rm_sem_deadlock([detail:int=5])
```

Parameters

pin_list
Returns a list of PINs that are deadlocked (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print deadlock information.

The default is 5. (optional parameter)

This macro will use a deadlock detection algorithm to find resource contention problems that concern semaphores (for native MPE/XL).

User Discussion

Example

```
$31c ($36) nmdat > rm_sem_deadlock
Getting semaphore info for ALL pins ...

 No Deadlock detected.
```

rm_sem_owner

Print (or return) the process that owns the specified semaphore.

Syntax

```
[pin_list:str] = rm_sem_owner (sem_addr:ptr,
                               [detail:int=5])
```

Parameters

pin_num
Returns the PIN of the process that owns the specified semaphore (optional parameter).

sem_addr
The virtual address of the semaphore that information is requested about.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print the process identification number of the process that owns the specified semaphore.

The default is 5. (optional parameter)

Prints and/or returns the owner pin of the process that owns the specified semaphore. If detail level is 0 then nothing is printed and the pin is returned.

User Discussion

Example

```
$31d ($36) nmdat > rm_sem_owner(00000001)

Owner of sharable semaphore at $00000001 is $0
```

rm_semaphore

Prints all the pertinent information regarding the specified semaphore.

Syntax

```
rm_semaphore([sem_addr:ptr=$0.0],
             [detail:int=5])
```

Parameters

sem_addr
The virtual address of the semaphore that information is requested about.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Prints the pertinent information regarding the specified semaphore.

The default is 5. (optional parameter)

Example

rm_semaphore_info

Shows the type, class, and semaphore type specific information for the specified semaphore.

Syntax

```
[field_info:any] = rm_semaphore_info(sem_addr:ptr=0.0],
                                     [detail:int=5],
                                     [field:str=''])
```

Parameters

field_info
Returns the specified field value (optional parameter).

sem_addr
The virtual address of the semaphore that information is requested about.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print the type, class, and semaphore type specific information for the specified semaphore.

The default is 5. (optional parameter)

field
If specified then this field will be retrieved from the semaphore record specified by sem_addr. Prints the type, class, and semaphore type specific information for the specified semaphore.

User Discussion

Example

rm_sir_blocked_proc

Print (or return) the processes (if any) that are blocked (impeded) on the specified SIR.

Syntax

```
pin_list:str = rm_sir_blocked_proc (sir_num:int,
                                    [detail:int=5])
```

Parameters

pin_list
Returns a list of PINs that are blocked on the specified SIR; the owner of the SIR is the first PIN in the list.

sir_num
The compatibility mode SIR index number.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print a list of process that are blocked on the specified SIR.

The default is 5. (optional parameter)

Prints out the owner and any processes that are blocked on a particular SIR. The returned list contains the owner PIN followed by the waiting PINs.

User Discussion

Example

```
$323 ($36) nmdat > wl rm_sir_blocked_proc(1,5)
```

rm_sir_deadlock

Indicates if any deadlocks are detected for compatibility mode SIRs.

Syntax

```
pin_list:str = rm_sir_deadlock (sir_num:int,
                                [detail:int=5])
```

Parameters

pin_list
Returns a list of pins that are deadlocked on compatibility mode SIRs.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print deadlock information.

The default is 5. (optional parameter)

This macro will use a deadlock detection algorithm to find resource contention problems that concern SIRs. (for compatibility mode).

User Discussion

Example

```
$324 ($36) nmdat > rm_sir_deadlock
Getting SIR info for ALL pins ...

 No Deadlock detected.
```

rm_sir_owner

Print (or return) the process that owns the specified SIR.

Syntax

```
pin_list:str = rm_sir_owner (sir_num:int,
                             [detail:int=5])
```

Parameters

pin_num
Returns the PIN of the process that owns the specified SIR (optional parameter).

sir_num
The SIR number that information is requested about. (required parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print the process identification number of the process that owns the specified SIR.

The default is 5. (optional parameter)

Prints and/or returns the owner pin of the process that owns the specified SIR. If detail level is 0 then nothing is printed and the pin is returned.

User Discussion

Example

```
$325 ($36) nmdat > rm_sir_owner(1)
SIR $1 : Load Process
   Owner PIN:  $0
   # of Waiting Processes:  $0
```

Contents

Chapter 9
User Interface Macros

9

User Interface Macros

Introduction

The UI macros include:

job/session management
command interpreter
break handling routine
reply information table

Since the command interpreter is not associated one-to-one with a job, all macros reference it by process id.

The user interface macros cover:

job/session management
command interpreter
break handling routine

The following abbreviations are used here for user interface tables:

CCB - Command Control Block (Command Interpreter)
JCUT - Job Cutoff Table (CM)
JDT - Job Directory Table (CM)
JIT - Job Information Table (CM)
JMAT - Job Master Table (CM)
JPCNT - Job Process Count Table (CM)
UIT - User Interrupt Table(?) (break table)

The term job as used in this manual always refers to either a batch job or a terminal session job.

The job number will be specified as a string (#Jxx or #Sxx).

ui_all_jobs

Returns a list of job/session numbers for all jobs and sessions currently active.

Syntax

```
job_list:str = ui_all_jobs
```

Parameters

job_list
: A list of the job numbers corresponding to all active jobs and sessions is returned here. This can be processed with the DAT FOREACH commands. (required parameter).

User Discussion

This macro does not return any errors unless the Job Master Table is mangled.

Macros Used

none

Example

```
$32a ($36) nmdat > wl ui_all_jobs
"#S1"
```

ui_capabilities

Prints the capabilities information associated with the specified job.

Syntax

```
ui_capabilities (job_num:str,
                 [detail:int=1],
                 [error_parm:str='pad'])
```

Parameters

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing.

1 - 5 Print job/session capabilities information.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

This is heavily environment dependent. Currently the allowed commands mask, capability set, and associated devices is printed for the specified job.

There are three things in the JIT that needs to be printed:

- Allow mask (allow_mask)
- Capabilities (user_attribute, general_resource)
- Associate table (assoc_table_ndx)

The allow mask is a bit mask which defines the operator commands this job is allowed to use see the MPE V tables manual pp 8-13 for the bit definitions..

In the file DTMPINCL.JOBSES there are type definitions called user_attrib_type and general_resources_type.

The associate table is handled just like in MPE V - see the MPE V tables manual for details.

Macros Used

Example

```
$32b ($36) nmdat > ui_capabilities('#s1')

ALLOW MASK : None

USER ATTRIBUTES :SM,AM,AL,GL,DI,OP,CV,UV,LG,SP,PS,NA,NM,CS,ND,SF,BA,IA,PM,MR,
                 DS,PH
```

ui_ccb

Prints the specified CCB table for the specified CI process and optionally returns the value of the specified field.

Syntax

```
return_val:any = ui_ccb ([pin_num:int=pin],
                         [ccb_type:str='kpo_process_ccb'],
                         [detail:int=1],
                         [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified. (optional parameter)

pin_num
The process identification number of the CI process to display the CCB for. The default is the current PIN. (optional parameter)

ccb_type
Specify the KPO name corresponding to the CCB to print. The default is kpo_process_ccb. (optional parameter)

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the table record type. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val. If the field_name is not specified, the entire table entry is formatted. (optional parameter)

User Discussion

The table is formatted using the symbolic formatter. The type of the return_val is determined by the type of the field_name specified.

Macros Used

```
kpo_number
kpo_pointer
```

Example

```
$32d ($36) nmdat > ui_ccb

PACKED RECORD
   CALLING_PROCESS          : CI
   PRINT_ERRORS             : ALL_MSGS
   SET_CIERROR_JCW          : TRUE
     ...                      .
     ...                      .
   DMYINT2                  : 0
   DMYPTR1                  : 0
   DMYPTR2                  : 0
END
```

ui_cicommand

Prints the Command Interpreter current command with the specified process.

Syntax

```
ui_cicommand ([pin_num:int=pin],
              [detail:int=1],
              [error_parm:str='pad'])
```

Parameters

pin_num
Specify the process identification number of the process to print the CI current command for. The default is the current pin. (optional parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print detailed information.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Macros Used

ui_ccb

Example

```
$330 ($36) nmdat > ui_cicommand
cmdline_addr :
40200138

RECORD
   LINE  : 'NEWGROUP lib ;cap=ph,ds,mr,pm,ia,ba;access=(r,x,l:any;a,w,s:gu)

   L_CNT : 1
   L_LEN : 40
   EOLS  :
      [ 1 ]: 40
      [ 2 ]: 0
      [ 3 ]: 0
      [ 4 ]: 0
      [ 5 ]: 0
      [ 6 ]: 0
      [ 7 ]: 0
      [ 8 ]: 0
      [ 9 ]: 0
      [ a ]: 0
END
```

ui_cihistory

Prints the Command Interpreter history stack associated with the specified process.

Syntax

```
ui_cihistory ([pin_num:int=pin],
                [num_entries:int=0],
                [detail:int=1],
                [error_parm:str='pad'])
```

Parameters

pin_num
Specify the process identification number of the process to print the CI history stack for. The default is the current pin. (optional parameter).

num_entries
Specify the number of entries to print starting from the last entry in the stack. The default is print all entries. (optional parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print process command interpreter history stack information.
	The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

The pointer to the redo stack is obtained by the ui_ccb macro (with the default CCB) using 'ccb_hdr.redo_stk_addr' From this, the redo stack can be printed. The redo stack is defined in DCIREDO.CI. Be sure and print only the number of entries requested, if that number is not zero.

Macros Used

ui_ccb

Example

ui_ciprocess

Returns a list of the command interpreter processes for the specified job.

Syntax

```
[pin_list:str] = ui_ciprocess (job_num:str,
                               [error_parm:str='pad'])
```

Parameters

pin_list
A list of the pins associated with the specified job is returned here. This can be processed with the DAT FOREACH commands. (optional parameter).

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

for each process returned by ui_process, check that the program file name (returned by pm_program) is equal to 'CI.PUB.SYS'.

Macros Used

```
ui_process
pm_program
```

Example

```
$334 ($36) nmdat > wl ui_ciprocess('#s1')

$36
```

ui_civariables

Prints the Command Interpreter variables associated with the specified process.

Syntax

```
ui_civariables (job_num:str,
                [detail:int=1],
                [error_parm:str='pad'])
```

Parameters

job_num
Specify the job number to print the CI variables for. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter)

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 - 5 Print the values of the variables.

The default is 1. (optional parameter)

Currently this parameter is not used.

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Each command interpreter variable is printed, along with the type and current value.

Macros Used

```
ui_jit
streadto
```

Example

```
$336 ($36) nmdat > ui_civariables('#s1')

Var #1        Name :CIERROR
Integer Value:      $3dd

Var #2        Name :HPACCOUNT
Function Addr:
RECORD
   GET : UNKNOWN
   PUT : UNKNOWN
END


  ...             ....
  ...             ....
Var #58       Name :STOREJCW
Integer Value:      $0

Var #59       Name :DEV
String Value:       tape:$
```

ui_devices

Prints all of the devices associated with the specified job.

Syntax

```
[ldev_list:str] = ui_devices (job_num:str,
                              [detail:int=1],
                              [error_parm:str='pad'])
```

Parameters

ldev_list
A list of the logical device numbers associated with the specified job is returned here. This list may be processed with the DAT FOREACH command. (optional parameter).

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing.

1 - 5 Print job/session device information.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

Example

```
$23d ($10) nmdat > ui_devices('#j2')

LDEV(s) associated with #j2 :  $7
```

ui_jdt

Prints the JDT entry associated with the specified job.

Syntax

```
[return_val:any] = ui_jdt (job_num:str,
                           [detail:int=1],
                           [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified. (optional parameter)

job_num
The job number of corresponding to the table entry to format. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the table record type. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val. If the field_name is not specified, the entire table entry is formatted. (optional parameter)

User Discussion

Perhaps some selectors should be allowed to format the different parts of the JDT.

Macros Used

ui_jmat

Example

ui_jit

Prints the JIT associated with the specified job.

Syntax

```
[return_val:any] = ui_jit (job_num:str,
                           [detail:int=1],
                           [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified. (optional parameter)

job_num
The job number of corresponding of the JIT format. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the table record type. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val. If the field_name is not specified, the entire table entry is formatted. (optional parameter)

User Discussion

Macros Used

none

Example

```
$33d ($36) nmdat > ui_jit('#s1')

CRUNCHED RECORD
   JIT_DST           : 85
   JIT_VALUE_6       : 6
   JIT_RESERVED_1    : 0
   JIT_VALUE_8       : 8
   .....
   .....
      [ e  ]: 0
      [ f  ]: ff
      [ 10 ]: ff
   PROCNAME          : '
END
```

ui_jmat

Prints the JMAT entry associated with the specified job.

Syntax

```
[return_val:any] = ui_jmat (job_num:str,
                            [detail:int=1],
                            [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified. (optional parameter)

job_num
The job number of corresponding to the table entry to format. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the table record type. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val. If the field_name is not specified, the entire table entry is formatted and the pointer to the JMAT entry is returned. (optional parameter)

User Discussion

Search the JMAT sequentially for the specified job number. The JMAT is the same CM DST as in MPE. entry_ptr is the pointer to the JMAT entry. The job/session number field is called JS_number. The number of entries is stored in entry zero. See the MPE V tables manual.

Macros Used

none

Example

```
$23c ($10) nmdat >ui_jmat('#j2',1)

CRUNCHED RECORD
   STATE             : 3
   DUPLICATIVE       : FALSE
   INTERACTIVE       : FALSE
   MESSAGE_STATE     : FALSE
   STDLIST_STATE     : FALSE
   JOB_ON_TERMINAL   : FALSE
   DEV_CLASS_INDEX   : TRUE
   INPRI             : b
   JS_TY             : 2
   JS_NUMBER         : 2
   JS_NUMBER_EXTEND  : 0
   USER_NAME         : 'MGR     '
   ACCOUNT_NAME      : 'RACHAL  '
   JOB_NAME          : 'RITTEST1'
   GROUP_NAME        : 'PUB     '
   INPUT_DEVICE      : 20
   OUTPUT_DEVICE     : 1f
   START_DATE        :
      YEAR : 57
      DAY  : f0
   START_TIME        :
      HOUR : e
      MIN  : 9
      SEC  : f
      MSEC : 6
   LANGUAGE          : 0
   EXEC_PRIORITY     : c8
```

```
   JSMAIN_PIN          : 20
   CPU_LIMIT           : 0
   SPOOLED             : TRUE
   RESTART             : FALSE
   SEQUENCED           : FALSE
   PSC                 : FALSE
   PSC_WAIT            : FALSE
   OUTPRI              : 8
   COPIES_NUMBER       : 1
   ORIGIN_JIN          : a
   ORIGIN_JLIST        : c
   JMAT_RESERVED_3     :
      [ 1 ]: 0
      [ 2 ]: 0
      [ 3 ]: 0
      [ 4 ]: 0
      [ 5 ]: 0
      [ 6 ]: 0
      [ 7 ]: 0
      [ 8 ]: 0
END
```

ui_job

Prints all information about a job.

Syntax

```
ui_job (job_num:str,
        [detail:int=1],
        [error_parm:str='pad'])
```

Parameters

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter). Call ui_all_job to get the job_num.

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0 Suppress printing.

1 - 5 Print information job/session.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

Macros Used

```
ui_process
ui_main_process
ui_devices
ui_ciprocess
ui_cicommand
ui_cihistory
ui_civariables
ui_user
ui_tempfiles
ui_schedule
ui_capabilities
```

Example

ui_job_port_msg

Prints the specified (or all) messages on the specified job port subqueue.

Syntax

```
[return_val:any] = ui_job_port_msg ([port_type:str='job_queue_port'],
                                    subqueue:int,
                                   [message_num:int=0],
                                   [detail:int=1],
                                   [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified and a single message is selected. (optional parameter)

port_type
Specify the name of the job port to format messages for:

jobmain_port	The JOB main port
job_queue_port	The job queue port
sessionmain_port	The SESSION main port

The default is job_queue_port. (optional parameter)

subqueue
Specify the subqueue number of the subqueue to look for the messages in. (required parameter)

message_num
Specify the number of the message in the subqueue. One indicates the first message on the queue. Specify zero (the default) to print all messages on the queue. (optional parameter)

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) message.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the message type declaration. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val if a specific message number is specified. If the field_name is not specified, the message is formatted. (optional parameter)

User Discussion

This macro is used for global job/session ports to determine what is waiting on the port. An error is returned if the job_port, message number, or subqueue number is invalid.

Macros Used

port_global

Example

ui_jpcnt

Prints the JPCNT number associated with the specified job.

Syntax

```
[jpcnt_num:int] = ui_jpcnt (job_num:str,
                            [detail:int=1])
```

Parameters

jpcnt_num
The index into the JPCNT for this job is returned here. (optional parameter)

job_num
The job number of corresponding to the JPCNT entry. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Display the JPCNT information.

The default is 1. (optional parameter)

User Discussion

The JPCNT entry number is returned. A message is displayed indicating if the bit in the JPCNT which corresponds to the entry number is set.

Macros Used

none

Example

ui_jsmain_port_msg

Prints the specified (or all) messages on the jsmain port.

Syntax

```
[return_val:any] = ui_job_port_msg (job_num:str,
                                    subqueue:int,
                                   [message_num:int=0],
                                   [detail:int=1],
                                   [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified and a single message is selected. (optional parameter)

job_num
Specify the job number to print the port messages for. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter)

subqueue
Specify the subqueue number of the subqueue to look for the messages in. (required parameter)

message_num
Specify the number of the message in the subqueue. One indicates the first message on the queue. Specify zero (the default) to print all messages on the queue. (optional parameter)

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) message.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the message type declaration. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val if a specific message number is specified. If the field_name is not specified, the message is formatted. (optional parameter)

User Discussion

An error is returned if the job number, message number, or subqueue number is invalid.

Macros Used

port_message

Example

ui_process

Prints the process tree associated with the specified job number.

Syntax

```
[pin_list:str] = ui_process (job_num:str,
                             [detail:int=1],
                             [error_parm:str='pad'])
```

Parameters

pin_list
A list of the pins associated with the specified job is returned here. This can be processed with the DAT FOREACH commands. (optional parameter).

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify a number between 0 and 5 which determines the level of d print as follows:

0 Suppress printing.

1 - 5 Print all process tree information associated with the specified job number.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

The process tree is printed by the pm_family macro.

Macros Used

```
ui_main_process
pm_family
```

Example

```
$348 ($35) nmdat > ui_process('#s1')

 $17

 PARENT PIN  PROGRAM FILE
 ----------  ------------
 $15         SESSION.PUB.SYS

 CHILD PIN   PROGRAM FILE
 ---------   ------------
 $36         CI.PUB.SYS

 JSMAIN PIN  PROGRAM FILE
 ----------  ------------
 $17         JSMAIN.PUB.SYS

 SIBLING PIN PROGRAM FILE
 ----------- ------------
 $19         JSMAIN.PUB.SYS
```

ui_reply

Prints a report of all replies pending.

Syntax

```
ui_reply ([detail:int=1])
```

Parameters

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print all replies pending information.

The default is 1. (optional parameter)

User Discussion

Macros Used

```
kso_number
kso_pointer
```

Example

ui_schedule

Prints the scheduling information associated with the specified job.

Syntax

```
ui_schedule (job_num:str,
             [detail:int=1],
             [error_parm:str='pad'])
```

Parameters

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print only summary level information.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

Prints the input priority, starting time and date, job state, executing priority, and output priority.

Macros Used

`ui_jmat`

Example

```
$348 ($36) nmdat > ui_schedule('#s1')
job state:              $1
input priority:         $8
executing priority:     $96
output priority:        $8
starting time:     HOUR: #11      MIN: #6       SEC: #23      TSEC: #3
starting date:     YEAR: #87      DAY: #163
```

ui_tempfiles

Prints any temporary files or file equations associated with the specified job.

Syntax

```
ui_tempfiles (job_num:str,
              [detail:int=1],
              [error_parm:str='pad'])
```

Parameters

job_num

Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail

Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print job/session temporary files or file equations.

The default is 1. (optional parameter)

error_parm

Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

File equations are currently not supported - they will be when the JDT is redesigned.

Macros Used

```
ui_main_process
fs_address
```

Example

```
$34a ($36) nmdat > ui_tempfiles('#s1')

No temp files
```

ui_uit

Formats the UIT for the specified pin.

Syntax

```
return_val:any = ui_uit  ([pin_num:int=pin],
                          [detail:int=1],
                          [field_name:str=''])
```

Parameters

return_val
The value of the field_name specified is returned here if a field_name is specified.

pin_num
The pin of the process that owns the break table to format. If omitted, the current process is assumed. (optional parameter).

detail
Specify one of the following values:

0	Do not print anything; just return the value.
1 - 5	Format the specified field in (or the entire) table entry.

The default is 1. (optional parameter)

field_name
Specify the field_name (as defined in the MODCAL type definition) of the desired field in the table record type. If the field_name is specified, only that field is formatted, and the value of that field is returned in return_val. If the field_name is not specified, the entire table entry is formatted. (optional parameter)

User Discussion

This table is formatted using the symbolic formatter. If the pin_num is not valid, an error will result.

Macros Used

pibx_info

Example

```
$34b ($36) nmdat > ui_uit

RECORD
   UIT_RBT_PTR              : c739ca00
   UIT_RBT_ENTRY_PTR        : c739cdc0
   UIT_NOTIFY_QUEUE         : ffffdfdd
   UIT_BREAK_QUEUE_DEPTH    : 0
   UIT_ABORT_QUEUE_DEPTH    : 0
   UIT_RESUME_QUEUE_DEPTH   : 0
   UIT_SSBRK_QUEUE_DEPTH    : 0
   UIT_IN_BREAK_MODE        : FALSE
   UIT_CAUSE_BREAK_PIN      : 7ffd
   UIT_SAVED_BREAK          : ENABLED
   UIT_SAVED_SSBRK          : DISABLED
   UIT_SAVED_ECHO           : FALSE
   UIT_SAVED_TIMEOUT        : 78
   UIT_SAVED_EOR            : '^

   UIT_SAVED_ALT_EOR        : ''
   UIT_SIGNAL_CHARS         : ''
   UIT_INTERRUPT_MASK       :
      SUBSYSTEM_BREAK : DISABLED
      BREAK           : ENABLED
      CAUSE_BREAK     : ENABLED
   UIT_BREAK_IS_STOLEN      : FALSE
   UIT_SAVED_STOLEN_BREAK   : DISABLED
   UIT_BREAK_RECEIVER       : 36
   UIT_BREAK_HANDLER        : UNKNOWN
   UIT_BREAK_INT_LEVEL      : 0
   UIT_BREAK_WAS_RECEIVED   : FALSE
   UIT_BREAK_MSG_PORT       : 0
   UIT_BREAK_MSG_SUBQUEUE   : 0
   UIT_SSBRK_IS_STOLEN      : FALSE
   UIT_SAVED_STOLEN_SSBRK   : DISABLED
   UIT_SSBRK_RECEIVER       : 7ffd
   UIT_SSBRK_HANDLER        :
      INT_EXEC_MODE : NM
         NM_INT_HANDLER : UNKNOWN
   UIT_SSBRK_INT_LEVEL      : 3
   UIT_SSBRK_WAS_RECEIVED   : FALSE
   UIT_SSBRK_MSG_PORT       : 0
   UIT_SSBRK_MSG_SUBQUEUE   : 0
   UIT_UDC_DISABLE_COUNT    : 0
   UIT_IN_A_JOB             : FALSE
   UIT_STDIN_LDEV           : 14
END
```

ui_user

Prints and returns the full user/account name associated with the specified job.

Syntax

```
user_name:str = ui_user ([job_num:str],
                         [detail:int=1],
                         [error_parm:str='pad'])
```

Parameters

user_name
Returns the user name in the standard user displayable format here. This includes the session name, user name, account name, and group name. (optional parameter).

job_num
Specify the job/session number of the job to process. See the job/session number display/specification format specified in the Job Management Function Section. (required parameter).

detail
Specify a number between 0 and 5 which determines the level of detail to print as follows:

0	Suppress printing.
1 - 5	Print job/session with user/account name information.

The default is 1. (optional parameter)

error_parm
Specify an error action to take when this macro, or a macro that this macro calls fails for any reason. See the standard error definition for details on how this is to be specified. The default is print an error message, add the message to the error stack, and terminate the macro. (optional parameter)

User Discussion

The user name is both displayed and returned in the format: session,user.account,group

Macros Used

ui_jmat

Example

```
$34c ($36) nmdat > ui_user('#s1')

 MANAGER.SYS,PUB
```

Contents

Chapter 10
Port Macros

10

Port Macros

port_info

This macro supplies port information.

Syntax

```
port_info
 ([port_num : S32 = $1] ,
  [pin_num  : U16 = $0] ,
  [msg_type : STR = ''] ,
  [detail   : INT / U16 = $0] )
```

Parameters

port_num

The port number to display information on. This number may be one of the Standard Local Process Ports (1, 2 or 3) or a Global Port (i.e. a negative number). The default is Process Local Port #1, the processes Standard Signal Port.

pin_num

This is the PIN number. This paramter is only used if a Standard Local Process Port is specified. If no PIN is specified, the current process PIN will be used.

msg_type

If there are any messages waiting at the port, this
message type will be used to format the messages. If no type string is supplied only the message pointers will be displayed. The macro will supply the msg_type string for the following known ports. These defaults may be overridden by supplying the msg_type string.

Local Port #3 - Standard Process Interrupt Port
Known Global Ports

pfp_port0	= -8
pfp_port1	= -27
pfp_port2	= -28
pfp_port3	= -29
sessionmain_port	= -12
jobmain_port	= -13
job_queue_port	= -14
sysmain_port	= -17

detail
Specify the level of printing.(optional parameter)

0 - 3	Format the messages using the default or supplied message type.
4 - 5	Display only the message pointers. No formatting of messages will be done.

Examples

```
$2bd ($1c) nmdat > port_info(-(#12))

Information for Port: $fffffff4

Port Type        : Message Port
Port is PFP      : FALSE
Purge Pending    : FALSE
Access Count     : $1
Owner Pin        : $17 (SESSION.PUB.SYS)
Ports PFP Port   : $ffffffe5
Std Ports Desc   : $00000000
Global Port Desc: $c70083d0
Freeze Desc      : $00000000

Subqueue Information

Max # Suqueues     : $12
Enabled Subq Mask : $ffff0fff
NonEmpty Subq Mask: $00002000
Dormant Subq Mask : $00000000
Number of Messages: $a

Server Information

Server Type        : Process Server
Server Invoked     : FALSE
Server Pin         : $17 (SESSION.PUB.SYS)

Message Pool Information

Use Count     : $1
Message Size : $274
Pool Desc     : $c0d0815c

Message Info:

Subqueue $12:


PACKED RECORD
   JS_REQUEST : 20
      JSM_PL_INFO :
         JSMAIN_PID  : 22b
         JSMAIN_PORT : ffffdfe8
END
```

```
$2bf ($1c) nmdat > port_info(3,25)

Information for Port: $fffffe81

Port Type        : Message Port
Port is PFP      : FALSE
Purge Pending    : FALSE
Access Count     : $0
Owner Pin        : $25 (JSMAIN.PUB.SYS)
Ports PFP Port   : $fffffff8
Std Ports Desc   : $c721cee8
Global Port Desc: $c700b230
Freeze Desc      : $00000000

Subqueue Information

Max # Suqueues     : $b
Enabled Subq Mask  : $fc000000
NonEmpty Subq Mask: $00000000
Dormant Subq Mask  : $00000000
Number of Messages: $0

Server Information

Server Type        : No Server
Server Invoked     : FALSE

Message Pool Information

Use Count      : $6e
Message Size : $80
Pool Desc      : $c70f0480
```

port_data

Returns the virtual address of the port data area.

Syntax

```
address:ptr = port_data (portid:int)
```

Parameters

address
The virtual address of the port data area is returned here. (required parameter)

portid
The port number of the port to return the address of the port data area for. (required parameter)

User Discussion

The virtual address of the port data area for the specified port is returned. An error is generated if the portid does not exist.

Macros Used

port_record

Example

port_global

Returns the port number for the specified global port name.

Syntax

```
portid:int = port_global (portname:str)
```

Parameters

portid
Returns the port identification here for the specified global port name. (required parameter)

portname
The port name constant as specified in the DKSPORT include file. (required parameter)

User Discussion

The port number is returned for the specified Known System Port name. An error is generated if the port name is not found.

Macros Used

none

Example

Contents

Chapter 11
Other Macros

Other Macros

11

cmport_record

Returns the virtual address of the CM port descriptor record associated with the specified port.

Syntax

```
address:ptr = cmport_record (portid:int)
```

Parameters

address
The virtual address of the CM port descriptor record for the specified port is returned here. (required parameter)

portid
The port number of the port to return the address of the descriptor record for. (required parameter)

User Discussion

An error is generated if the portid does not exist.

Macros Used

```
kso_number
kso_pointer
```

Example

footprint_global

Formats the specified global footprint entry.

Syntax

```
next_entry = footprint_global (entry_num:int,
                               [header : bool=true])
```

Parameters

entry_num
The entry number of the footprint entry to be formatted. (required parameter)

header
If true then a table header will be printed with the entry information. Default value is true. (optional parameter)

User Discussion

This macro can be used to format the global footprint entry only. The entry number of the next valis footprint entry in the table is returned. If the entry specified is the last entry in the table, -1 is returned.

Example

subsysstr

Returns a subsystem string mnemonic based on the number passed in.

Syntax

```
subsysname:str = subsysstr(subsysnum:int)
```

Parameters

subsysnum
The subsystem number that you want the name of.

subsysname
The subsystem that you want the name of.

Example

```
$ef ($3) nmdat> wl subsysstr(#122)
Port (IPC) Facility
```

sys_vuf

Prints or returns the system's Version-Update-Fix level.

Syntax

```
sys_vuf ([detail=1])
```

Parameters

detail
Specify a number to determine the level of print. The default is 0. (optional parameter)

0	detail will suppress printing.
1 - 5	will print the desired field. (optional parameter)

field_name
The name of the field requested.

User Discussion

This macro obtains the system's VUF from the CM sysglob area.

Example

```
$328 ($36) nmdat > sys_vuf

 9.5..6C
```

sysglob

Prints or returns the specified field from the system globals area

Syntax

```
sysglob (detail=1, field_name='')
```

Parameters

detail

Specify a number to determine the level of print. The default is 0. (optional parameter)

0	detail will suppress printing.
1 - 5	will print the desired field. (optional parameter)

User Discussion

This macro can be used to retrieve any field form the system globals area.

Example

```
$36c ($36) nmdat > wl sysglob('memory_size')

$4000

$36c ($36) nmdat > wl sysglob('system_state')

SYSTEM_UP_STATE
```

HEWLETT PACKARD
Part Number32650-90094
Edition 1
E1287
Printed in U.S.A.