UCRL-15336

P.O. 3634909

Praxis Input/Output Interface

Richard E. Shapiro
Arthur Evans, Jr.
Bolt Beranek and Newman Inc.

MASTER

January 1981

BBN Report No. 4583

PRAXIS INPUT/OUTPUT INTERFACE

Richard E. Shapiro
Arthur Evans, Jr.

January 1981

Prepared by Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02238

Performed for Lawrence Livermore National Laboratory
University of California

Contract 3634909

DISCLAIMER

This book was prepared as a document of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability for or represents any of the accuracy, completeness, or usefulness of any information, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or approval by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

	Page
1. GENERAL	1
2. DECLARING RMS FACILITIES -- PDP11 ONLY	3
3. INITIALIZING RMS -- PDP11 ONLY	5
4. RMS CONTROL BLOCKS	7
5. FIELDS IN CONTROL BLOCKS	13
5.1 Integers	13
5.2 Enumerations	14
5.3 Bit strings	14
6. FILE OPERATIONS	15
7. RECORD OPERATIONS	19
8. COMPLETION ROUTINES AND ERROR HANDLING	25
9. LIST OF PROCEDURES AND CONSTANTS	27
9.1 VAX constants	27
9.1.1 access constants [module IOPTYP]	27
9.1.2 access_mode constants [module IOPTYP]	27
9.1.3 alignment_boundary_type constants [module IOPATR]	27

9.1.4	allocation option constants [module IOPATR]	27
9.1.5	characteristic constants [module IOPTYP]	28
9.1.6	file organization constants [module IOPTYP]	28
9.1.7	file processing option constants [module IOPTYP]	28
9.1.8	key option constants [module IOPATR]	29
9.1.9	key type constants [module IOPATR]	29
9.1.10	Protection code constants [module IOPATR]	29
9.1.11	record attribute constants [module IOPTYP]	30
9.1.12	record format constants [module IOPTYP]	30
9.1.13	record processing option constants [module IOPTYP]	30
9.1.14	sharing mode constants [module IOPTYP]	31
9.2	RSX constants	31
9.2.1	access constants [module IOPTYP]	31
9.2.2	access mode constants [module IOPTYP]	31
9.2.3	alignment boundary type constants [module IOPATR]	31
9.2.4	allocation option constants [module IOPATR]	32
9.2.5	characteristic constants [module IOPTYP]	32
9.2.6	file organization constants [module IOPTYP]	32
9.2.7	file processing option constants [module IOPTYP]	32
9.2.8	key option constants [module IOPATR]	32
9.2.9	key type constants [module IOPATR]	33
9.2.10	protection code constants [module IOPATR]	33
9.2.11	record attribute constants [module IOPTYP]	33
9.2.12	record format constants [module IOPTYP]	33
9.2.13	record processing option constants [module IOPTYP]	33
9.2.14	sharing mode constants [module IOPTYP]	34
9.3	Other Types	34
9.4	Procedures	35

CHAPTER 1: GENERAL

This document is intended as an introduction to the use of RMS facilities via Praxis (this interface hereafter called Praxis-RMS). It is presumed that the reader is familiar with Praxis conventions as well as with RMS use (at the MACRO level). Since Praxis-RMS was designed to be functionally equivalent to MACRO-RMS, the explanations follow the pattern of the DEC MACRO-RMS documentation (particularly the programmer's reference manual).

A complete list of the procedures that make up Praxis-RMS appears at the end of this document (with parameters), along with the constants (grouped by type) that can be used as actual parameters. Most of these procedures are also mentioned in one of the topically oriented chapters 2 through 7.

CHAPTER 2: DECLARING RMS FACILITIES -- PDP11 ONLY

As in the MACRO interface to RMS, you must declare the kinds of operations you wish to perform (the ORG\$ macro) and allocate space for buffers and internal RMS control structures (POOL macros). Since these declarations must be made at assembly time, they are not included as part of the Praxis-RMS interface. Instead, a small MACRO module should be written to include the ORG\$ and POOL macros. This module is then linked to the Praxis-coded modules at task-build time.

CHAPTER 3: INITIALIZING RMS -- PDP11 ONLY

To initialize RMS (\$INIT in MACRO interface) you should call the Praxis procedure `Init_RMS()`. This actually does the \$INITIF macro. You must call this procedure in order to do any subsequent RMS operations.

CHAPTER 4: RMS CONTROL BLOCKS

All parameters to RMS are passed via control blocks. The four basic types of blocks are:

1. File Access Block (FAB) for file operations
2. Record Access Block (RAB) for record operations
3. Extended Attribute Blocks (XAB's) for extra parameters to file operations
4. File Name Block (NAM) for detailed file-name descriptions

For each of these kinds of blocks there is a (hidden) structure declaration in Praxis-RMS. Made available to the user are pointers to these hidden structures as follows:

1. The pointer type File points to FAB's
2. The pointer type Stream points to RAB's
3. The pointer type Attribute points to XAB's
4. The pointer type File_Name points to NAM's

This package was written before starting and finishing were implemented in Praxis. Now that they are, the package should be modified to use them so that declaring any variables to be of one of these types will automatically allocate a structure and fill in the essential fields. Until that is done, the STARTING routines must be called explicitly to do the necessary initialisation. These routines are:

1. procedure New_file (file_id :out ref File)
2. procedure New_stream (stream_id : out ref Stream)
3. procedure New_attribute (attribute_id : out ref Attribute)

4. procedure New_Name (file_name_id :out ref file_name)

Similarly, the FINISHING routines must also be called explicitly. These will free the structure and perform simple cleanup operations (e.g. disconnecting a stream or closing a file). The procedures are:

1. procedure Rel_File (file_id : out ref File)
2. procedure Rel_Stream (stream_id : out ref Stream)
3. procedure Rel_Attribute (attribute_id : out ref Attribute)
4. procedure Rel_Name (file_name_id :out ref File_Name)

After allocating a File or Stream, the fields within the associated control blocks can only be modified by passing appropriate parameters to the various File and Record operation procedures.

For File_Name's and the various Attribute's the fields within the block are explicitly filled in by the user. There is one procedure for each of these blocks which takes the (allocated-initialised) pointer to the block and any user-specifiable field-values as parameters. These procedures are:

```
procedure Mk_Name
```

```
  param
```

```
    file_name_id : File_name
```

```
    expanded_string_buffer :in ref array [1..?e] of char
```

```
    resultant_string_buffer : in ref array [1..?r] of char
```

```
    related_file_name : File_Name
```

```
  endparam
```

```
procedure Mk_Date_Time_Block
```

```
  param
```

```
    attribute_id : attribute
```

```
    creation_time : encoded_time
```

```
    revision_time : encoded_time
```

```
    revision_number : ioword
```

```
    next_block : attribute
```

```
  endparam
```

```
procedure Mk_File_Protection_Block
```

```
  param
    attribute_id : attribute
    user_id : in ref UIC
    f_protection : in ref protection
    next_block : attribute
  endparam
```

```
procedure Mk_Allocation_Cntrl_Blk
```

```
  param
    attribute_id : attribute
    boundary_type : alignment_boundary_type
    allocation : iolong
    options : allocation_option
    bucket_size : iobyte
    def_extension : ioword
    al_location : iolong
    volume : ioword
    next_block : attribute
  endparam
```

```
procedure Mk_Key_Definition_Block
```

```
  param
    attribute_id : attribute
    data_bucket_area_number : iobyte
    data_bucket_fill_size : iobyte
    data_type : key_type
    options : key_option
    index_bucket_area_number : iobyte
    index_bucket_fill_size : ioword
    key_buffer : in ref array [1..?k] of char
    lowest_index_level_area_number : iobyte
    null_key : char
    key_position : in ref array [0..7] of ioword
    ref_number : iobyte
    key_size : in ref array [0..7] of iobyte
    next_block : attribute
  endparam
```

```
procedure Mk_Summary_Block
```

```
  param
    attribute_id : attribute
    next_block : attribute
  endparam
```

```

procedure Mk_Revision_Block // VAX only
  param
    attribute_id : attribute
    revision_date : encoded_time
    revision_num : ioword
    next_block : attribute
  endparam

procedure Mk_File_Header_Block // VAX only
  param
    attribute_id : attribute
    next_block : attribute
  endparam

```

Note that the last two procedures are for VAX only.

To use an Attribute for any operation, it must be set up by one of the above procedures. These fill in essential RMS info as well as the user-specified field values. In addition, each of these has a Next_attribute parameter which allows for the linking of several XAR's to be used on a single file operation.

For each user-accessible field in any of the control-blocks, a function will be provided to return the value of that field. This is the only way to get the contents of a field.

1. function rec_access (stream_id : stream) returns answer : access_mode
2. function rec_length (stream_id : stream) returns answer : ioword
3. function rec_status (stream_id : stream) returns answer : status
4. function fil_access (file_id : file) returns answer : access
5. function fil_attr (file_id : file) returns answer : record_attribute
6. function fil_format (file_id : file) returns answer : record_format

7. function fil_maxsiz (file_id : file) returns answer : ioword
8. function fil_org (file_id : file) returns answer :
file_organization
9. function fil_share (file_id : file) returns answer : sharing_mode
10. function fil_status (file_id : file) returns answer : status

CHAPTER 5: FIELDS IN CONTROL BLOCKS

In general, there are three sorts of fields--simple integers (e.g. the default-extension field of a FAB); enumerations, in which one of a predefined group of values goes in the field (e.g. the organization field of a FAB--one of Indexed, Relative, or Sequential); and bit-strings, in which several of a predefined group of options go in the field (e.g. the record-access field of FABs which can take one or more of Put, Get, Delete, etc.). Each of these has a different manner of specification/examination, detailed below.

5.1 Integers

Integers in Praxis-RMS structures are one of four types:

1. iobyte (8 bits)
2. ioword (16 bits)
3. iolong (32 bits)
4. integer (i.e. machine dependent)

To specify a value, or compare to a given value, you should use constants and variables of the appropriate type for the field in question.

Enumerations

5.2 Enumerations

For each enumeration field, there is a group of legal constants exported by Praxis-RMS. No other values should be used either to specify or compare with enumeration fields.

5.3 Bit strings

For each bit-string field, a group of option-constants is exported by Praxis-RMS. Only logical combinations (i.e., **ands**, **ors**, and **nots**) of these should be used to specify or compare with bit-string fields. When sets are implemented in Praxis, bit-string fields should be implemented as sets.

CHAPTER 6: FILE OPERATIONS

Praxis-RMS file operations work by filling the FAB with the specified actual parameters, and invoking the appropriate RMS macro.

The `error_proc` and `success_proc` parameters do not have corresponding fields in the FAB. Their use is explained below in Chapter 8.

procedure RMS_Create

```
  param
    file_id : File
    allowed_access : access
    name_string : in ref array [1..?n] of char
    name_len : integer
    format : record format
    organization : File organization
    attribute_list : optional attribute initially nil
    file_name_id : optional File Name initially nil
    options : optional file_processing_option initially 8#0
    allocation : optional iolong initially 0
    bucket_size : optional iobyte initially 0
    block_size : optional ioword initially 0
    def_extension : optional ioword initially 0
    def_name_string : optional in ref array[1..?d] of char initially ""
    def_len : optional integer initially 0
    fixed_control : optional iobyte initially 0
    max_rec_num : optional iolong initially 0
    max_rec_size : optional ioword initially 0
    newattributes : optional record attribute initially 8#0
    window_size : optional iobyte initially 8#0
    how_share : optional sharing_mode initially 8#0
    channel : optional iobyte initially 0 //PDP11 only
    error_proc : optional fcmp_prc initially fno_cmp
    success_proc : optional fcmp_prc initially fno_cmp
  endparam
```

procedure RMS_Open

param

file_id : File

allowed_access : access

name_len : integer

name_string : in ref array [1..?n] of char

options : optional file_processing_option initially 8#0

attribute_list : optional attribute initially nil

file_name_id : optional File_Name initially nil

def_extension : optional ioword initially 0

def_name_string : optional in ref array [1..?d] of char initially ""

def_len : optional integer initially 0

window_size : optional iobyte initially 0

how_share : optional sharing_mode initially 8#0

channel : optional iobyte initially 0 //PDP11 only

error_proc : optional fcmp_prc initially fno_cmp

success_proc : optional fcmp_prc initially fno_cmp

endparam

procedure RMS_Dsplay

param

file_id : File

attribute_list : optional attribute initially nil

error_proc : optional fcmp_prc initially fno_cmp

success_proc : optional fcmp_prc initially fno_cmp

endparam

procedure RMS_Erase

param

file_id : File

name_string : optional in ref array [1..?n] of char initially ""

name_len : integer

file_name_id : optional File_Name initially nil

options : optional file_processing_option initially 8#0

def_name_string : optional in ref array [1..?d] of char initially ""

def_len : optional integer initially 0

channel : optional iobyte initially 0 //PDP11 only

error_proc : optional fcmp_prc initially fno_cmp

success_proc : optional fcmp_prc initially fno_cmp

endparam


```
procedure RMS_Extend
```

```
  param
```

```
    file_id : File
```

```
    options :optional file_processing_option initially 8#0
```

```
    attribute_list :optional attribute initially nil
```

```
    allocation :optional iolong initially 0
```

```
    error_proc :optional fcmp_prc initially fno_cmp
```

```
    success_proc :optional fcmp_prc initially fno_cmp
```

```
  endparam
```

```
procedure RMS_Close
```

```
  param
```

```
    file_id : File
```

```
    options :optional file_processing_option initially 8#0
```

```
    error_proc :optional fcmp_prc initially fno_cmp
```

```
    success_proc :optional fcmp_prc initially fno_cmp
```

```
  endparam
```

CHAPTER 7: RECORD OPERATIONS

Praxis-RMS record operations work by filling the RAB with the specified actual parameters, and invoking the appropriate RMS macro.

The `error_proc` and `success_proc` parameters do not have corresponding fields in the RAB. Their use is explained below in Chapter 8.

Note that in the Find, Get, and Put operations there are formal parameters `key_buffer` (an array of char) and `key_number` (pointer to an integer). These are for random-by-key access of index and random files, respectively. If a non-nil `key_number` is specified, it has priority. Therefore, if you're doing random-by-key access of an indexed file, the `key_number` actual should be nil, and the `key_buffer` actual a string; if you're doing random-by-key access of a random file, the `key_number` actual should point to a number (and the `key_buffer` actual is ignored). If you're not doing random-by-key access, then these are both dummies.

Note also the `prompt_buffer` formal parameter in the Get operation (VAX only). Recall that the prompt-address field and the key-address field of a RAB are coincident. Since a prompt-buffer is only relevant for terminal I/O, and since terminals are not indexed files, there is no logical conflict. The physical conflict of which actual parameter sets the RAB field is resolved as follows: If the prompt option is specified (as a bit in the options parameter) then `prompt_buffer` is used, otherwise `key_buffer` is used.

procedure RMS_Connect

```

  param
    stream_id : Stream
    file_id : File
    reference_key : optional iobyte initially 0
    options : optional record_processing_option initially 8#0
    multi_block_count : optional iobyte initially 0
    multi_buffer_count : optional iobyte initially 0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam

```

procedure RMS_Disconnect

```

  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam

```

procedure RMS_Find

```

  param
    stream_id : Stream
    record_access_mode : access mode
    key_buffer : optional array [1..?k] of char initially ""
    key_len : optional integer initially 0
    key_number : optional pointer iolong initially nil
    reference_key : optional iobyte initially 0
    options : optional record_processing_option initially 8#0
    rfa : optional array [1..3] of ioword
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam

```

procedure RMS_Get

```

param
    stream_id : Stream
    record_access_mode : access_mode
    user_buffer : in ref array [1..?u] of char
    user_len : integer
    reference_key : optional iobyte initially 0
    key_buffer : optional in ref array [1..?b] of char initially ""
    key_len : optional integer initially 0
    key_number : optional pointer iolong initially nil
    options : optional record_processing_option initially 8#0
    rfa : optional array [1..3] of ioword
    header_buffer : optional in ref array [1..?h] of char initially ""
    prompt_buffer : optional in ref array [1..?p] of char initially ""
    //VAX only
    prompt_len : optional integer initially 0 //VAX only
    time_out : optional iobyte //VAX only
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
endparam

```

procedure RMS_Put

```

param
    stream_id : Stream
    record_access_mode : access_mode
    record_buffer : in ref array [1..?r] of char
    rec_len : integer
    key_buffer : optional in ref array [1..?k] of char initially ""
    key_len : optional integer initially 0
    key_number : optional pointer iolong initially nil
    options : optional record_processing_option initially 8#0
    header_address : optional in ref array [1..?h] of char initially ""
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
endparam

```

procedure RMS_Update

```

param
    stream_id : Stream
    record_address : in ref array [1..?r] of char
    rec_len : integer
    options : optional record_processing_option initially 8#0
    header_address : optional in ref array [1..?h] of char initially ""
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
endparam

```

procedure RMS_Delete

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
```

procedure RMS_Rewind

```
  param
    stream_id : Stream
    reference_key : optional iobyte initially 0
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
```

procedure RMS_Truncate

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
```

procedure RMS_Flush

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
```

procedure RMS_NextVolume

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
```

CHAPTER 8: COMPLETION ROUTINES AND ERROR HANDLING

All of the Praxis-RMS operations allow the specification of completion routines for successful and/or error operation-completion. These are Praxis routines of the appropriate type (see below) that work essentially like standard RMS completion routines. The difference is that they have a parameter (file or stream, depending on the operation).

File-operation completion routines must be of type

`fcmp_prc` is procedure (`file_id` : file)

If you do not specify a completion routine (i.e. you use the default) then no completion routine will be invoked. Recall that some implementations of PDP11 RMS do not use success-completion routines (though they are not illegal to specify).

Stream-operation completion routines must be of type

`scmp_prc` is procedure (`stream_id` : stream)

If you do not specify a completion routine (i.e. you use the default) then no completion routine will be invoked.

Completion routines are tied in with Praxis-RMS error handling. Before any operation, a special boolean field adjacent to the control block is set to true, indicating that the block is busy. Any completion routines transparently reset this field. If the operation was successful and no success-completion routine was specified then the "busy" field is reset by the Praxis-RMS operation. If the operation is not successful, and no error-completion routine was specified, then the field is reset, and an exception is raised.

Since the status of an asynchronous operation is indeterminate at the time of return from RMS, no checking of the above sort is done. Therefore, both completion routines must be specified for asynchronous operations. Failure to do so is considered an error, and an exception is raised. Any operation with a control-block whose "busy" field is set is also considered an error (causing an exception).

Any exceptions that occur in (or below) an asynchronous completion-routine will abort the program.

CHAPTER 9: LIST OF PROCEDURES AND CONSTANTS

9.1 VAX constants

9.1.1 access constants [module IOPTYP]

- delete_access
- get_access
- put_access
- truncate_access
- update_access
- block_only_access
- block_access

9.1.2 access_mode constants [module IOPTYP]

- sequential_access
- random_by_key
- random_by_RFA

9.1.3 alignment_boundary_type constants [module IOPATR]

- cylinder
- logical_block
- virtual_block

9.1.4 allocation_option constants [module IOPATR]

- contiguous
- hard
- on_cylinder_boundary
- contiguous_best_try

9.1.5 characteristic constants [module IOPTYP]

carriage_control_device
directory_structured_device
record_oriented_device
single_directory_device
block_oriented_device
terminal
device_allocated
device_available
device_dismount
device_error_log
file_oriented_device
input_device
mail_box
device_mounted
output_device
random_access_device
real_time_device
shareable_device
device_spooled
device_write_locked

9.1.6 file_organization constants [module IOPTYP]

relative
indexed
sequential

9.1.7 file_processing_option constants [module IOPTYP]

contiguous
make_delete_on_close
use_File_Name
not_EOF
current_position
rewind_on_close
rewind_on_open
supersede
temporary
mark_temporary
contiguous_best_try
create_if

- deferred_write
- escape
- sysinput
- known_open
- maximize_version
- nonfile_structured
- output_file_parse
- process_permanent
- read_check
- submit_command_file
- spool_on_close
- sequential_only
- truncate_on_eof
- user_file_mode
- user_file_open
- write_check

9.1.8 key_option constants [module IOPATR]

- can_change
- duplicate_allowed
- null_value

9.1.9 key_type constants [module IOPATR]

- string_key
- signed_2_byte_key
- unsigned_2_byte_key
- signed_4_byte_key
- unsigned_4_byte_key
- packed_decimal

9.1.10 Protection_code constants [module IOPATR]

- no_read
- no_write
- no_execute
- no_delete

9.1.11 record_attribute constants [module IOPTYP]

block_contained
carriage_return
Fortran_format
print_format

9.1.12 record_format constants [module IOPTYP]

fixed
variable
variable_fixed_control
undefined_format

9.1.13 record_processing_option constants [module IOPTYP]

asynchronous
position_eof
locate_mode
key_greater_or_equal
key_greater
load_by_fill
update_if
block_io
cancel_ctrl_o
convert_lower
limit_key
no_lock
nonexistent_processing
prompt
purge_typeahead
read_ahead
allow_read_locked
read_no_echo
read_no_filter
timeout
truncate_put
manual_unlock
write_behind

9.1.14 sharing_mode constants [module IOPTYP]

put_share
get_share
delete_share
update_share
no_share
multi_stream
user_interlock

9.2 RSX constants

9.2.1 access constants [module IOPTYP]

delete_access
get_access
put_access
read_access
truncate_access
write_access
update_access

9.2.2 access_mode constants [module IOPTYP]

sequential_access
random_by_key
random_by_RFA

9.2.3 alignment_boundary_type constants [module IOPATR]

cylinder
logical_block
virtual_block

9.2.4 allocation_option constants [module IOPATR]

contiguous
hard

9.2.5 characteristic constants [module IOPTYP]

carriage_control_device
directory_structured_device
record_oriented_device
single_directory_device
block_oriented_device
terminal

9.2.6 file_organization constants [module IOPTYP]

relative
indexed
sequential

9.2.7 file_processing_option constants [module IOPTYP]

contiguous
make_delete_on_close
unlock_abnormal
use_File_Name
not_EOF
current_position
rewind_on_close
rewind_on_open
supersede
temporary
mark_temporary

9.2.8 key_option constants [module IOPATR]

can_change
duplicate_allowed
null_value

9.2.9 key_type constants [module IOPATR]

string_key
signed_2_byte_key
unsigned_2_byte_key
signed_4_byte_key
unsigned_4_byte_key
packed_decimal

9.2.10 protection_code constants [module IOPATR]

no_read
no_write
no_execute
no_delete

9.2.11 record_attribute constants [module IOPTYP]

block_contained
carriage_return
Fortran_format

9.2.12 record_format constants [module IOPTYP]

fixed
variable
variable_fixed_control
stream_format
undefined_format

9.2.13 record_processing_option constants [module IOPTYP]

asynchronous
position_eof
locate_mode
key_greater_or_equal
key_greater
load_by_fill
update_if

```

mass_insert
fast_delete

```

9.2.14 sharing_mode constants [module IOPTYP]

```

write_share

```

9.3 Other Types

```

attribute is    pointer internal_attribute [module IOPATR]
encoded_time is array [1..4] of ioword [module IOPATR]
fcmp_prc is     procedure (file_id : file) [module IOPTYP]
file is         pointer internal_file [module IOPTYP]
file_name is    pointer internal_file_name [module IOPNAM]
iobyte is       8 bit integer [module IOPMSC]
iolong is       32 bit integer [module IOPMSC]
ioword is       16 bit integer [module IOPMSC]

protection is
  packed structure [module IOPATR]
    system : protection_code
    owner : protection_code
    group : protection_code
    world : protection_code
  endstructure

scmp_prc is     procedure (stream_id : stream) [module IOPTYP]

status is
  packed structure [module IOPTYP]
    status_code : integer
    status_value : integer
  endstructure

stream is       pointer internal_stream [module IOPTYP]

```

```
UIC is
  packed structure [module IOPATR]
    programmer_number : ioword
    project_number : ioword
  endstructure
```

9.4 Procedures

```
function fil_access (file_id : file)
  returns answer : access [module IOPTYP]

function fil_attr (file_id : file)
  returns answer : record_attribute [module IOPTYP]

function fil_format (file_id : file)
  returns answer : record_format [module IOPTYP]

function fil_maxsiz (file_id : file)
  returns answer : ioword [module IOPTYP]

function fil_org (file_id : file)
  returns answer : file_organization [module IOPTYP]

function fil_share (file_id : file)
  returns answer : sharing_mode [module IOPTYP]

function fil_status (file_id : file)
  returns answer : status [module IOPTYP]

procedure Init_RMS () [module RMSINIT--PDP11 only]

procedure Mk_Allocation_Cntrl_Blz
  param
    attribute_id : attribute
    boundary_type : alignment_boundary_type
    allocation : iolong
    options : allocation_option
    bucket_size : iobyte
    def_extension : ioword
    al_location : iolong
    volume : ioword
    next_block : attribute
  endparam
  [module IOPATR]
```



```

procedure Mk_Date_Time_Block
  param
    attribute_id : attribute
    creation_time : encoded_time
    revision_time : encoded_time
    revision_number : ioword
    next_block : attribute
  endparam
  [module IOPATR]

procedure Mk_File_Header_Block
  param
    attribute_id : attribute
    next_block : attribute
  endparam
  [module IOPATR--VAX only]

procedure Mk_File_Protection_Block
  param
    attribute_id : attribute
    user_id : in ref UIC
    f_protection : in ref protection
    next_block : attribute
  endparam
  [module IOPATR]

procedure Mk_Key_Definition_Block
  param
    attribute_id : attribute
    data_bucket_area_number : iobyte
    data_bucket_fill_size : iobyte
    data_type : key_type
    options : key_option
    index_bucket_area_number : iobyte
    index_bucket_fill_size : ioword
    key_buffer : in ref array [1..?k] of char
    lowest_index_level_area_number : iobyte
    null_key : char
    key_position : in ref array [0..7] of ioword
    ref_number : iobyte
    key_size : in ref array [0..7] of iobyte
    next_block : attribute
  endparam
  [module IOPATR]

```

```
procedure Mk_Name
  param
    file_name_id : File_name
    expanded_string_buffer : in ref array [1..?e] of char
    resultant_string_buffer : in ref array [1..?r] of char
    related_file_name : File_Name
  endparam
  [module IOPNAM]

procedure Mk_Revision_Block
  param
    attribute_id : attribute
    revision_date : encoded_time
    revision_num : ioword
    next_block : attribute
  endparam
  [module IOPATR--VAX only]

procedure Mk_Summary_Block
  param
    attribute_id : attribute
    next_block : attribute
  endparam
  [module IOPATR]

procedure New_attribute (attribute_id : out ref Attribute)
  [module IOPATR]

procedure New_file (file_id : out ref File)
  [module FILENV]

procedure New_Name (file_name_id : out ref File_Name)
  [module IOPNAM]

procedure New_stream (stream_id : out ref Stream)
  [module STRENV]

function rec_access (stream_id : stream)
  returns answer : access_mode [module IOPTYP]

function rec_length (stream_id : stream)
  returns answer : ioword [module IOPTYP]

function rec_status (stream_id : stream)
  returns answer : status [module IOPTYP]
```

```
procedure Rel_Attribute (attribute_id : out ref Attribute)
    [module IOPATR]
```

```
procedure Rel_File (file_id : out ref File)
    [module FILENV]
```

```
procedure Rel_Name (file_Name_id : out ref File_Name)
    [module IOPNAM]
```

```
procedure Rel_Stream (stream_id : out ref Stream)
    [module STRENV]
```

```
procedure RMS_Close
    param
        file_id : File
        options : optional file_processing_option initially 8#0
        error_proc : optional fcmp_prc initially fno_cmp
        success_proc : optional fcmp_prc initially fno_cmp
    endparam
    [module CLOS]
```

```
procedure RMS_Connect
    param
        stream_id : Stream
        file_id : File
        reference_key : optional iobyte initially 0
        options : optional record_processing_option initially 8#0
        multi_block_count : optional iobyte initially 0
        multi_buffer_count : optional iobyte initially 0
        error_proc : optional scmp_prc initially sno_cmp
        success_proc : optional scmp_prc initially sno_cmp
    endparam
    [module CONN]
```

procedure RMS_Create

```
  param
    file_id : File
    allowed_access : access
    name_string : in ref array [1..?n] of char
    name_len : integer
    format : record format
    organization : file organization
    attribute_list : optional attribute initially nil
    file_name_id : optional File_Name initially nil
    options : optional file_processing_option initially 8#0
    allocation : optional iolong initially 0
    bucket_size : optional iobyte initially 0
    block_size : optional ioword initially 0
    def_extension : optional ioword initially 0
    def_name_string : optional in ref array [1..?d] of char initially ""
    def_len : optional integer initially 0
    fixed_control : optional iobyte initially 0
    max_rec_num : optional iolong initially 0
    max_rec_size : optional ioword initially 0
    newattributes : optional record_attribute initially 8#0
    window_size : optional iobyte initially 8#0
    how_share : optional sharing_mode initially 8#0
    channel : optional iobyte initially 0 //PDP11 only
    error_proc : optional fcmp_prc initially fno_cmp
    success_proc : optional fcmp_prc initially fno_cmp
  endparam
  [module CREA]
```

procedure RMS_Delete

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
  [module DELE]
```

procedure RMS_Disconnect

```
  param
    stream_id : Stream
    options : optional record_processing_option initially 8#0
    error_proc : optional scmp_prc initially sno_cmp
    success_proc : optional scmp_prc initially sno_cmp
  endparam
  [module DISC]
```

```
procedure RMS_Display
```

```
  param
```

```
    file_id : File
```

```
    attribute_list : optional attribute initially nil
```

```
    error_proc : optional fcmp_prc initially fno_cmp
```

```
    success_proc : optional fcmp_prc initially fno_cmp
```

```
  endparam
```

```
    [module DISP]
```

```
procedure RMS_Erase
```

```
  param
```

```
    file_id : File
```

```
    name_string : optional in ref array [1..?n] of char initially ""
```

```
    name_len : integer
```

```
    file_name_id : optional File_Name initially nil
```

```
    options : optional file_processing_option initially 8#0
```

```
    def_name_string : optional in ref array [1..?d] of char initially ""
```

```
    def_len : optional integer initially 0
```

```
    channel : optional iobyte initially 0 //PDP11 only
```

```
    error_proc : optional fcmp_prc initially fno_cmp
```

```
    success_proc : optional fcmp_prc initially fno_cmp
```

```
  endparam
```

```
    [module ERAS]
```

```
procedure RMS_Extend
```

```
  param
```

```
    file_id : File
```

```
    options : optional file_processing_option initially 8#0
```

```
    attribute_list : optional attribute initially nil
```

```
    allocation : optional iolong initially 0
```

```
    error_proc : optional fcmp_prc initially fno_cmp
```

```
    success_proc : optional fcmp_prc initially fno_cmp
```

```
  endparam
```

```
    [module EXTE]
```

Procedures

```
procedure RMS_Find
```

```
  param
```

```
    stream_id : Stream
```

```
    record_access_mode : access_mode
```

```
    key_buffer : optional array [1..?k] of char initially ""
```

```
    key_len : optional integer initially 0
```

```
    key_number : optional pointer iolong initially nil
```

```
    reference_key : optional iobyte initially 0
```

```
    options : optional record_processing_option initially 8#0
```

```
    rfa : optional array [1..3] of ioword
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module FIND]
```

```
procedure RMS_Flush
```

```
  param
```

```
    stream_id : Stream
```

```
    options : optional record_processing_option initially 8#0
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module FLUS]
```

```
procedure RMS_Free
```

```
  param
```

```
    stream_id : Stream
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module FREE]
```

```
procedure RMS_Get
```

```
  param
```

```
    stream_id : Stream
```

```
    record_access_mode : access mode
```

```
    user_buffer : in ref array [1..?u] of char
```

```
    user_len : integer
```

```
    reference_key : optional iobyte initially 0
```

```
    key_buffer : optional in ref array [1..?b] of char initially ""
```

```
    key_len : optional integer initially 0
```

```
    key_number : optional pointer iolong initially nil
```

```
    options : optional record_processing_option initially 8#0
```

```
    rfa : optional array [1..3] of ioword
```

```
    header_buffer : optional in ref array [1..?h] of char initially ""
```

```
    prompt_buffer : optional in ref array [1..?p] of char initially ""
```

```
    //VAX only
```

```
    prompt_len : optional integer initially 0 //VAX only
```

```
    time_out : optional iobyte //VAX only
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
endparam
```

```
  [module GET]
```

```
procedure RMS_NextVolume
```

```
  param
```

```
    stream_id : Stream
```

```
    options : optional record_processing_option initially 8#0
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
endparam
```

```
  [module NEXT]
```

Procedures

```
procedure RMS_Open
```

```
  param
```

```
    file_id : File
```

```
    allowed_access : access
```

```
    name_len : integer
```

```
    name_string : in ref array [1..?n] of char
```

```
    options : optional file_processing_option initially 8#0
```

```
    attribute_list : optional attribute initially nil
```

```
    file_name_id : optional File_Name initially nil
```

```
    def_extension : optional ioword initially 0
```

```
    def_name_string : optional in ref array [1..?d] of char initially ""
```

```
    def_len : optional integer initially 0
```

```
    window_size : optional iobyte initially 0
```

```
    how_share : optional sharing_mode initially 8#0
```

```
    channel : optional iobyte initially 0 //PDP11 only
```

```
    error_proc : optional fcmp_prc initially fno_cmp
```

```
    success_proc : optional fcmp_prc initially fno_cmp
```

```
  endparam
```

```
    [module OPEN]
```

```
procedure RMS_Put
```

```
  param
```

```
    stream_id : Stream
```

```
    record_access_mode : access_mode
```

```
    record_buffer : in ref array [1..?r] of char
```

```
    rec_len : integer
```

```
    key_buffer : optional in ref array [1..?k] of char initially ""
```

```
    key_len : optional integer initially 0
```

```
    key_number : optional pointer iolong initially nil
```

```
    options : optional record_processing_option initially 8#0
```

```
    header_address : optional in ref array [1..?h] of char initially ""
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module PUT]
```

```
procedure RMS_Rewind
```

```
  param
```

```
    stream_id : Stream
```

```
    reference_key : optional iobyte initially 0
```

```
    options : optional record_processing_option initially 8#0
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module TRUN]
```



```
procedure RMS_Truncate
```

```
  param
```

```
    stream_id : Stream
```

```
    options : optional record_processing_option initially 8#0
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module TRUN]
```

```
procedure RMS_Update
```

```
  param
```

```
    stream_id : Stream
```

```
    record_address : in ref array [1..?r] of char
```

```
    rec_len : integer
```

```
    options : optional record_processing_option initially 8#0
```

```
    header_address : optional in ref array [1..?h] of char initially ""
```

```
    error_proc : optional scmp_prc initially sno_cmp
```

```
    success_proc : optional scmp_prc initially sno_cmp
```

```
  endparam
```

```
    [module UPDA]
```