UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 80

(March 1987)

Title:

EMAS-3: Subsystem Language Independent Programming Interface

Author: Keith Yarwood

Contact: Your Support Team

Software Support Category: n/a

Synopsis

This Note describes the specifications for the procedures comprising the Subsystem interface of EMAS-3.

Keywords

EMAS3 procedures, S# procedures, Subsystem commands, Subsystem data structures, Subsystem error messages, Subsystem interface

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

This document gives the specifications for the procedures comprising the Subsystem interface of EMAS-3. The first section describes the language-independent interface (procedures with names commencing "EMAS3"). The second section gives the Subsystem commands, which are accessible through the interface procedures EMAS3 and EMAS3H. The third section gives the interface procedures suitable for use by IMP programs in particular (entry point names commencing "S#"). The "S#" procedures are not documented in this second release, but the use of many of them can be deduced by inspection of the corresponding "EMAS3" entry (if any) from Section 1. These procedures are provided for use by IMP programs, which will execute with marginally fewer CPU cycles and page turns than would be the case using the "EMAS3" procedures. The "S#" procedures have the further slight advantage that the programmer is better able to judge which parameters are input and which are output parameters to the procedures, since results cannot be returned in value-type parameters. In addition, the "S#" procedures are in many cases close in specification to corresponding procedures on EMAS 2900. The fourth section gives the important Subsystem Data Structures, again incomplete in this second release, but initially comprising file formats.

In each case, where a %integername parameter FLAG is indicated, this is an "output" variable whose value after the call indicates the result of the operation invoked: zero indicates success; non-zero indicates reason for failure. A text string describing the failure can be obtained by calling EMAS3FAILUREMESSAGE or S#FAILUREMESSAGE. If the procedure EMAS3SETRETURNCODE or S#RETURNCODE is called with FLAG as the parameter, the explanatory message will be printed at the interactive terminal and the value of FLAG will be made available to the next program to be executed via procedures EMAS3RETURNCODE or S#RETURNCODE. If the value of FLAG is negated before EMAS3SETRETURNCODE or S#RETURNCODE is called, then the value will be made available to the next program to be executed, but the message will not be printed.

All command and procedure entrypoints to the Subsystem which may be of use to programmers are listed in file SUBSYS:EMAS3SPECS on EMAS-A.
The fifth section gives Subsystem Error Messages. The Subsystem error messages are also available in file SUBSYS:ERRORTEXTS.

1. The Language independent interface

All the 'EMAS3' procedures are %external %routines and have name parameters. The pairs of letters given as comments after each set of parameters indicate whether the parameters are "read" or "write" parameters for the procedure. For example {srswibiw} indicates that the first parameter {sr} is of type string (character) and will be read but not written by the procedure. The second (string) parameter will not be read, but will be written by the procedure. The third (integer) parameter will be both read and written by the procedure. The fourth parameter is of type integer and will be written, but not read, by the procedure.

```
%externalroutinespec EMAS3 %c
    (%stringname command, params, %integername flag)        {srsriw}
```

The command interpreter is invoked to load and execute the command COMMAND using parameters PARAMS. The command-line history mechanism is not invoked (either for retrieval or storage of COMMAND). If the command invoked is not part of the subsystem, the i/o channel statuses and file-connection statuses of the calling program are protected against interference by the called command. The string supplied in parameter COMMAND is prefixed with "C#" by the command interpreter before the search is made for the required entry point.

%externalroutinespec EMAS3ALLOWINTERRUPTS

This procedure is called after invocation of the Subsystem contingency arrangements, when an event has been trapped with further asynchronous events inhibited (see procedure EMAS3SETTRAP). Asynchronous events are again allowed to interrupt the process after this procedure has been called.

%externalroutinespec EMAS3CHANGEACCESS %c
(%stringname file, %integername mode, flag) {sririw}

The current connection mode of FILE is changed to MODE, provided that the caller's access permission allows. MODE may take values between 1 and 5. Bit values of MODE have the following meanings:

1 read access
2 write access
4 shared access.

Read access is always implied with write or shared access. If the connection mode of the file is already as requested, the call succeeds.

%externalroutinespec EMAS3CHANGEFILESIZE %c
(%stringname file, %integername newsize, flag) {sririw}

The physical size of FILE is altered to NEWSIZE, provided, for an increase in size, that there is sufficient filespace available to the caller and, if the file is currently connected in the caller's virtual memory, provided that there is enough space in the caller's virtual memory commencing at the current connect address. If FILE already has physical size equal to NEWSIZE, the call is successful. NEWSIZE is the number of BYTES required as the new size, and is rounded up before allocation or de-allocation to the next multiple of 4096 bytes. The third word (word 2) of the file header in *not* altered by this procedure. It is the caller's responsibility to ensure that this word correctly reflects the file's physical size if necessary.

%externalroutinespec EMAS3CHECKNAME %c
(%stringname name, %integername type, qualifier, flag) {sbiririw}

This procedure is used to establish the correctness and attributes of a filename. It also produces the full filename in parameter NAME. (The full filename may differ from the initial value of NAME because NAME can be given as a localname which is subject to possible prefixing by ownername and one or more groupnames. The parameter NAME is checked according to the specification in TYPE and QUALIFIER, and the result is given in FLAG. FLAG zero means that NAME matches the specification; non-zero values describe the mismatch. If a "USEGROUP" is in force, the "usegroup" string is prefixed to NAME before the checks are made.

TYPE may take *one* of the following values:

1 a file name or a partitioned file member name is acceptable
2 file name is acceptable
4 group name is acceptable
8 fileindex name is acceptable
16 username name is acceptable
32 path name is acceptable

QUALIFIER is also bit significant and is used to establish the existence, access and file type, as follows:

1	can be read
2	can be written
4	already exists
8	does not exist
16	do not add suffix to T# name

If the file *does* already exist, its type must be:

32	object file	(file type 1)
64	non-standard	(file type 2)
128	character file	(file type 3)
256	data filetype	(file type 4)
512	corrupt object	(file type 5)
1024	pdfile	(file type 6)
2048	directory	(file type 7)

%externalroutinespec EMAS3CLAIMCHANNEL (%integername chan) {iw}

This procedure provides a currently unused i/o channel number for use in EMAS3DEFINE and subsequent i/o operations. Channel numbers supplied by this procedure are automatically closed and the definitions cleared on return to Subsystem command level: programs need not trouble to perform these operations if they are not logically required by the program itself.

%externalroutinespec EMAS3CLEARJOURNAL

This procedure clears the "recall" file for the process.

%externalroutinespec EMAS3CLOSE (%integername channel, flag) {iriw}

When you wish to read from or write to a device or a file the sequence of events that must be followed is:

1. associate a channel number with the device or file (see EMAS3DEFINE and EMAS3CLAIMCHANNEL);
2. the channel is 'opened'. This is often done implicitly by the first read or write operation;
3. when reading or writing is complete, the channel is closed by this routine;
4. the definition of the channel is cleared so that the channel may be re-used.

%externalroutinespec EMAS3COMREG (%integername n, value) {iriw}

This procedure supplies in VALUE the current contents of word N of the Subsystem internal communication region comprising the integer array COMREG(0:60). The following table gives the use of these COMREG words.

1 to 5	Unused
6	Initialisation indicator for FORTRAN i/o
7	Count of unsatisfied entries during loading
8	Reserved
9	Unused
10	Reserved
11	Address of table for EBCDIC to ISO code conversion

12	Address of table for ISO to EBCDIC code conversion			
13	Reserved			
14	Address of workfile used by compilers and object-code generation procedures.			
15 to 18	Unused			
19	Reserved			
20 to 21	Unused			
22	Current input channel number			
23	Current output channel number			
24	Value supplied to last call of EMAS3SETRETURNCODE or S#SETRETURNCODE			
25	Switch providing extra diagnostics from Subsystem routines			
26	Unused			
27 with 28	Bits set by the command PARM and used by compilers and linker, as follows:			
27	Bit value	Mnemonic	Bit value	Mnemonic
	2**0	QUOTES	2**16	OPT
	2**1	NOLIST	2**17	MAP
	2**2	NODIAG	2**18	DEBUG
	2**3	STACK	2**19	FREE
	2**4	NOCHECK	2**20	DYNAMIC
	2**5	NOARRAY	2**21	diag stream set
	2**6	NOTRACE	2**22	EBCDIC
	2**7	PROFILE	2**23	NOLINE
	2**8	NORANGE	2**24	stack size set
	2**9	INHIBIOF	2**25	NOMAIN
	2**10	ZERO	2**26	PARMZ
	2**11	XREF	2**27	PARMY
	2**12	LABELS	2**28	PARMX
	2**13	LET	2**29	MISMATCH
	2**14	CODE	2**30	Unused
	2**15	ATTR	2**31	Unused
28	2**0	Unused	2**16	Unused
	2**1	I8	2**17	Unused
	2**2	F8	2**18	Unused
	2**3	R8	2**19	Unused
	2**4	OPTEXT	2**20	OPT1
	2**5	NOCOMMENTS	2**21	OPT2
	2**6	NOWARNINGS	2**22	OPT3
	2**7	STRICT	2**23	OPT4
	2**8	MAXDICT	2**24	Unused
	2**9	Unused	2**25	Unused
	2**10	Unused	2**26	Unused
	2**11	Unused	2**27	Unused
	2**12	Unused	2**28	Unused
	2**13	Unused	2**29	Unused
	2**14	MINSTACK	2**30	Unused
	2**15	Unused	2**31	Unused
29 to 34	Unused			
35	Reserved			
36	Address of stack display for return to command level			
37	Reserved			
38	Address of user GLA			
39	Loader parameters, set by command LOADPARM			
	Bit value	Mnemonic		
	2**0	MIN		
	2**1	LET		
	2**2	ZERO		
40	Channel number for compiler error messages			
41 to 43	Unused			

44	Address of first free byte in user GLA (used by the loader)
45	Unused
46	Reserved
47	Reserved
48	Reserved
49	Unused
50	Reserved
51	Unused
52	Address of object filename
53	Address of listing filename
54 to 57	Unused
58	Reserved
59 to 60	Unused

```
%externalroutinespec EMAS3CONNECT (%stringname file,
   %integername mode, hole, prot, conad, type, start, end, flag)    {srirririribiwiwiwiw}
```

This procedure connects, and supplies the connect address of, the file or pdfile member FILE. MODE must be set to the required connect mode required. The following bits are significant; the others are unused and reserved:

Bit value	Effect
x'00000001'	read mode
x'00000002'	write mode
x'00000004'	execute
x'00000008'	accept shared write
x'00000010'	new-copy

Common valid combinations of bits used in MODE are (in decimal):

1	read
2 or 3	read/write
9	read (accept shared write)
11	read/write (accept shared write)
19	read/write/newcopy

Note: new-copy mode is used when the existing content of FILE is of no further interest. This mode provides an *efficient* way of presenting a completely zero file.

If the MODE parameter contains the "write" bit, the file header Word 5 is set to the current date and time, in the format described in the section on File Formats.

HOLE provides a way of indicating the maximum expected size of FILE, after possible later calls of CHANGEFILESIZE. It is used to reserve a hole of sufficient size in the virtual memory for later extension of the file. HOLE is specified as a number of bytes, but if zero is specified a default value equal to the size of FILE rounded up to the machine's segment size is taken (1 Mbyte for IBM/XA implementations of EMAS).

PROT again is bit significant. It should be set zero unless one of the following effects is required:

Bit value	Effect
x'00000002'	available for immediate disconnection (for example when the virtual memory hole is required for another connect).
x'00000004'	soft-permanent connection (the Subsystem will not allow disconnection of this file during the current session).
x'00000008'	hard-permanent connection (Director will not allow disconnection of this file during the current session).
x'00000040'	the parameter CONAD is set with the required connect address.

x'00000080' the left-most byte of PROT contains the number of the FSYS (file system) containing FILE

The parameters CONAD, FILETYPE, DATASTART, DATAEND need not be set before the call of CONNECT, and are set in the event of successful connection as follows. CONAD is set to the connect address (segment-aligned, that is a multiple of 1 Mbyte, in the case of a file as opposed to a pdfile member). FILETYPE, DATASTART and DATAEND are taken from the file header (see section on Subsystem Data Structures). Briefly, FILETYPE is the content of word CONAD+12, being:

1 for an object file
2 for a non-standard file, not having a type value between 1 and 8, other than 2. In the case of FILETYPE=2, this will *not* normally be the content of word CONAD+12.
3 for a character file
4 for a data file
5 for a corrupt object file (after a failed compilation)
6 for a pdfile (partitioned file)
7 for a directory file
8 for a recall ("journal") file

DATASTART, DATAEND are the contents of words at CONAD+4, CONAD+0 respectively and indicate the offset in bytes of the start and end of useful data within the allocated phiscal size of the file (a multiple of 4096 bytes).

%externalroutinespec EMAS3CPUTIME (%longrealname time) {w}

This procedure provides in the parameter TIME the CPU time used by the process, measured in seconds, since the beginning of the session.

%externalroutinespec EMAS3DATE (%stringname date) {sw}

This procedure returns today's date, in the form dd/mm/yy, in the string (character) variable offered as parameter DATE.

%externalroutinespec EMAS3DEFINE %c
(%stringname identifier, %integername channel, flag) {sririw}

Associates a channel number with either a device or a file. See also EMAS3CLAIMCHANNEL.

%externalroutinespec EMAS3DESTROY (%stringname file, %integername flag) {sriw}

The file, or pdfile member, FILE is destroyed, provided that it is not in use by other processes, or by a program which has called the calling program, and provided that it has not been previously marked for permanent connection during the current session. FILE may or may not be currently connected in the caller's virtual memory. If it *is* connected, it is desirable that it *not* be disconnected before EMAS3DESTROY is called because the disconnection can be achieved without the necessity for page-out disc transfers.

%externalroutinespec EMAS3DISCARDTRAP (%integername id, flag) {iriw}

This procedure is used to discard a contingency trap, having identifier supplied in parameter ID. Details of the mechanism are given in User Note 65.

%externalroutinespec EMAS3DISCONNECT (%stringname file, %integername flag){sriw}

The file FILE is disconnected from the virtual memory, provided that it is not in use by a program which has called the calling program, and provided that it has not been marked for permanent connection during the session.

%externalroutinespec EMAS3ENTER %c
(%integername mode, code, gla, ep, pword, %stringname params) {iriririrsr}

EMAS3ENTER is used to enter a procedure previously loaded by a call of EMAS3LOADEP. MODE specifies whether or not a stack-switch is required; it can normally be set zero. CODE, GLA, EP and PWORD are four information words returned by the call of LOADEP, and provide sufficient information for the entry to the required procedure to be effected. PARAMS is used when the called procedure takes an IMP %string(255) parameter, and contains the parameter string to be passed to the called procedure.

EMAS3LOADEP and EMAS3ENTER should not normally be used for the dynamic loading and execution of commands. The procedure EMAS3 (q.v.) is preferred because the features and advantages of the command interpreter are thereby invoked, and the calling program is afforded a degree of protection of its files and i/o channels.

%externalroutinespec EMAS3ERROR (%stringname text) {sr}

This routine writes a message to the terminal to say where it was called from together with the parameter text. In addition, a monitor dump is written to the file S#ERROR.

%externalroutinespec EMAS3ETOI (%integername addr, length) {irir}

Converts length bytes of data starting at addr from EBCDIC to ISO.

%externalroutinespec EMAS3EVENTDATA (%integername firstdata,flag) {iwiw}

This procedure returns an array of fifty 32-bit words describing the process state at the most recent contingency. Successive words are as follows:

Name	Content	
CLASS	The class of the contingency	
	0	floating underflow
	1	floating overflow
	2	fixed overflow
	3	decimal overflow
	4	zero divide
	5	bound check
	6	size error
	7	B overflow
	8	stack error
	9	privilege ("address error")
	10	specification exception
	11	segment table format (should not occur)
	12	operation exception
	13	data exception
	16	system call
	17	cpu time exceeded

	18	disc transfer fail
	19	block wrong length
	20	hard CPU or store fault
	21	illegal OUT
	22	local controller error
	32	virtual store error
	64	interval timer
	65	single character `Int:` from terminal (SUBCLASS = ISO value of the single character)
	66	text message
	67	off stack top
	68	message from another process invocation
SUBCLASS	supplies extra information for some values of CLASS	
PSW0	program status word zero (defined in the "Principles of Operation" for the host machine).	
PSW1	program status word one (the instruction address)	
GR(0:15)	values of the machine's General Registers 0:15	
FR(0:3)	values of the machine's Floating Point registers 0:3	
CR(0:15)	values of the machine's Control Registers 0:15	
TIMER0	value of the process timer zero	
TIMER1	value of the process timer one	
ASYNCINH	zero if the process state was interruptible by asynchronous events; one if the process state was non-interruptible.	
PEANDILC	information about the last program error or SVC instruction	
SPARE		
TIMEOFDAY	this triple-word contains the time of day of the last updating of the state record, in the form of an IMP string.	

%externalroutinespec EMAS3EXIST (%stringname file, %integername flag) {sriw}

FLAG is set zero if FILE does not exist (or if the caller has no permission to access FILE), one if FILE does exist and the caller has at least read-access.

%externalroutinespec EMAS3EXISTTYPE (%stringname file, %integer name flag) {sriw}

FLAG is set zero if FILE does not exist (or if the caller has no permission to access FILE), otherwise it is set to the TYPE of the file:

1	for an object file
2	for a file whose type is not defined in the standard subsystem ("non-standard")
3	for a character file
4	for a data file
5	for a corrupt object file (after a failed compilation)
6	for a pdfile (partitioned file)
7	for a directory file
8	for a recall ("journal") file

%externalroutinespec EMAS3FAILUREMESSAGE %c
(%integername n, %stringname text) {irsw}

The character (string) variable passed as parameter TEXT is set with the failure message corresponding to the Subsystem error flag provided in parameter N.

%externalroutinespec EMAS3FILL (%integername length, from, filler) {iririr}

LENGTH bytes of virtual storage from address FROM are set equal to the value of the rightmost byte of FILLER.

%externalroutinespec EMAS3GETJOURNAL (%stringname file, %integername flag){swiw}

This procedure generates a file containing a "recall" of the current and previous interactive sessions (limited by the size of the circular file reserved to this purpose). The parameter FILE is set with the name of the character file produced.

%externalroutinespec EMAS3GETRESTOFLINE (%stringname line) {sw}

This procedure provides the calling program with those characters, from the parameters supplied on the command-line typed to invoke the program, which have not been "consumed" by earlier calls on PAM procedures. More details are given in Chapter 18 of the EMAS-3 User's Guide.

%externalroutinespec EMAS3GIVEEVENT (%integername class, subclass) {iwiw}

This procedure returns the CLASS and SUBCLASS of the most recent contingency and is normally called following invocation of the TRAP mechanism, described in User Note 65.

%externalroutinespec EMAS3H %c
(%stringname command, params, %integername flag) {srsriw}

The command interpreter is invoked to load and execute the command COMMAND using parameters PARAMS. The command-line history mechanism is invoked (both for retrieval or storage of COMMAND). The i/o channel statuses and file-connection statuses of the calling program are protected against interference by the called command. The string supplied in parameter COMMAND is prefixed with "C#" by the command interpreter before the search is made for the required entry point.

%externalroutinespec EMAS3HELP INFO (%stringname info) {sr}

This is an alternative way of setting the help field required by PAM. It is necessary only when a command has no parameters and the command writer wishes to provide help information in response to ??.

%externalroutinespec EMAS3HTOS (%integername i, places, %stringname s) {irirsw}

The 32-bit value of parameter I is represented as a string of up to eight hexadecimal digits. The right-most PLACES of the string are returned in parameter S.

%externalroutinespec EMAS3INCREMENT (%integername variable, by, res) {iwiriw}

This is used to adjust the value of a variable in a 'safe' way, i.e. even when it is being updated simultaneously by several processes. The value returned in RES is the original value of the variable.

%externalroutinespec EMAS3INPOS (%integername pos) {iw}

The procedure returns the position in the current input line of the character which will be delivered by the next call of (IMP) READCH, or equivalent.

%externalroutinespec EMAS3INTEGER (%stringname qualifier, %integername value){sriw}

This procedure produces an integer parameter value as described in User Note 62 on the parameter acquisition mechanism (PAM).

%externalroutinespec EMAS3INTERRUPT (%stringname interrupt) {sw}

This procedure provides the multiple-character string (if any) which has been typed in response to an `Int:` prompt at the interactive terminal. A null string is provided if no such interrupt message has been received.

%externalroutinespec EMAS3ITOE (%integername addr, length) {irir}

Converts length bytes of data starting at addr from ISO to EBCDIC.

%externalroutinespec EMAS3ITOS (%integername i, %stringname s) {irsw}

This procedure returns in S a string representing the value in decimal of the integer I. [Etymology: Integer **TO** String.]

%externalroutinespec EMAS3JOURNALOFF

The placement of interactive-terminal data into the "recall" file is inhibited following a call of this procedure.

%externalroutinespec EMAS3JOURNALON

If the placement of interactive-terminal data into the "recall" file was inhibited by a previous call of EMAS3JOURNALOFF, then such placement is uninhibited.

%externalroutinespec EMAS3LASTPARAM (%stringname text) {sw}

Gives the actual text that was used to derive the value for the most recently acquired parameter. See Chapter 17 of the EMAS-3 User's Guide for further details.

%externalroutinespec EMAS3LOADEP (%stringname entry, actualepname,
%integername type, flag, code, gla, ep, pword, level) {srswibiwiwiwiwiwir}

This procedure invokes the Subsystem loader to perform a cascade load for entrypoint ENTRY. The actual entrypoint found may differ from that requested because ENTRY may be an alias for some other entrypoint name. ACTUALEPNAME returns the actual entrypoint found. TYPE must be set to a combination of bit-values 1 and 2, indicating respectively data-type and code-type entrypoint names. The actual type of the entrypoint found is returned in parameter TYPE.

FLAG indicates the success or otherwise of the procedure-call, and CODE, GLA, EP, PWORD are returned as four values which are to be used to ENTER the loaded

program. See procedure EMAS3ENTER. LEVEL is an input parameter used to indicate whether or not the load is to be permanent (LEVEL=0) or at the current load level (LEVEL=1). Further details about the EMAS-3 Subsystem Loader are given in User Note 85.

```
%externalroutinespec EMAS3LONGREAL (%stringname qualifier,
    %longrealname value)                {srlw}
```

This procedure produces an longreal parameter value as described in User Note 62 on the parameter acquisition mechanism (PAM).

```
%externalroutinespec EMAS3MODPDFILE %c
    (%integername ep, %stringname pdfile, member, infile, %integername flag)
                                                                {irsrsrsriw}
```

This procedure provides services for the manipulation of partitioned files, as follows.

EP =	1	Copy file INFILE to filemember PDFILE_MEMBER, which must *not* already exist
	2	Destroy PDFILE_MEMBER
	3	Rename PDFILE_MEMBER so that its new name is PDFILE_INFILE
	4	Create (empty) partitioned file PDFILE
	5	Tidy the partitioned file: remove unused space between members and arrange for the pdfile index to have a modicum of spare space for future members to be added.

```
%externalroutinespec EMAS3MONPARAMS (%stringname template)   {sr}
```

This routine performs monitoring of the parameters passed to the routine calling it. A template of the parameters of the calling routine is supplied, in the same order as they appear in its spec, so that monparams can find them on the stack. As some parameters are only of interest on entry to a routine and others only on exit, a means of specifying this is given. As well as parameters, a text string supplied by the calling program can be added to the output.

The parameters are defined in monparam's TEMPLATE parameter. This takes the form:

<direction> [<type print>] [*<comments>]

DIRECTION is encoded as one character and indicates whether monparams is being called on entry to the calling routine or just before exit from it. The value '>' indicates a call on entry (requesting display of input params significant on entry); '<' is used where monparams is called just before exit from the routine to be monitored (display output params). Alternatively, where only a single call is to be made (rather than both at entry and exit), or where a one-off message is to be displayed, the direction should be specified as '='.

TYPE/PRINT is encoded as a pair of characters. The first character defines TYPE and the second defines whether the parameter is input only, output only, or set both at input and output. For each parameter of the calling routine, a pair of TYPE/PRINT characters is specified.

A COMMENT can be added to the end of the template (perhaps to indicate the result of a function) following the character pairs. This is defined by giving '*' in place of the TYPE character. The comment is printed following the parameter monitoring.

In more detail, the DIRECTION character is one of the following:

> for a call on entry
< for a call on exit
= where only a single call is made

The first character of each successive pair defines TYPE:

s for a stringname (IMP only)
t for a string(255) (IMP only)
u for a string(31) (IMP only)
i for an integername
j for an integer

The second character of each successive pair defines whether the parameter is PRINTed on this call:

'r' for an input only param ('>')
'w' for an output only param ('<' or '=')
'b' for a param both input and output ('>' or '<' or '=')

The printing produced by calls of EMAS3MONPARAMS is controlled by use of the special Subsystem command

```
Command:#NOTE n
```

By default no printing is produced. A call of #NOTE with non-zero parameter initiates the desired monitor printing. The parameter value for #NOTE is bit-significant, with bit-allocations as follows:

1 output to terminal (0 = to recall file only,provided that a RECALL option has been specified using the OPTION command)
2 reserved for application programs
4 Subsystem procedures are monitored, especially those concerned with file-handling (connect, disconnect)
8 Other Subsystem procedures are monitored, many being very frequently called
16 the name of the calling routine is printed (one routine-level back)
32 the name of the previous calling routine is printed (i.e. two routine-levels back)

It can be seen that

```
Command:#NOTE 3
```

is appropriate to invoke console printing monitoring entries to application programs using EMAS3MONPARAMS. Monitoring is disabled by typing

```
Command:#NOTE 0
```

or

```
Command:#N
```

Examples:

```
%externalroutine x(%string(255) s, %integername flag)
   %if ssown_ssnote&64#0 %then monparams(">triw")
      .
      .
      .
   %if ssown_ssnote&64#0 %then monparams("<triw")
%end
```

or,

```
monparams("<triw*res=".itos(res))
```

or,

```
monparams("=triw")
```

Switch on monitoring with #note; the bigger the number, the more you get:

```
Command:#note 7
```

is usually plenty.

%externalroutinespec EMAS3MOVE (%integername length, from, to) {iririr}

LENGTH bytes of data are moved from virtual address FROM to virtual address TO. The effect is null for negative and zero values of LENGTH. The procedure cannot be used to propagate a pattern through virtual memory, since it is coded to treat overlapping "from" and "to" areas as though they do not overlap.

%externalroutinespec EMAS3MOVESTRING (%stringname from, to,
%integername flag) {srswiw}

Performs a translation on a string so that it may be passed between programs written in different languages. The flag is zero if the translation is successful.

%externalroutinespec EMAS3NEWGEN (%stringname FILE1, FILE2,
%integername flag) {srsriw}

This procedure provides a mechanism for updating a file even though it may be currently in use (connected in one or more other users' virtual memories). Such files cannot be destroyed, renamed or overwritten in the ordinary way. FILE1 and FILE2 must both exist. FILE1 contains data which are to be the "new generation" of data for the file named FILE2.

To reduce the likelihood of accidental destruction of wanted data in FILE2 through provision of incorrect filenames, the procedure checks that the types (held in the fourth word of the file headers) of FILE1 and FILE2 are identical.

After a successful call of this procedure, the file named FILE1 "disappears" from the file index of the process owner. The file named FILE2 contains, to all intents and purposes, the contents of the former file FILE1. The former contents of FILE2 are invisible to all *new* connections of FILE2. The contents of FILE1 have become the "new generation" of data in FILE2.

If FILE2 was connected in other virtual memories at the time of the call, those virtual memories continue to "see" the former contents of FILE2. All *new* connections of FILE2 thereafter "see" only the new contents of FILE2. When FILE2 is disconnected from the last of the virtual memories in which it was connected at the time EMAS3NEWGEN was called, the former contents of FILE2 are lost for ever. If FILE2 was not connected in any other virtual memory at the time of the call, the transfer of the new generation of data takes places immediately.

EMAS3NEWGEN attempts to disconnect both FILE1 and FILE2 from the caller's virtual memory before the above operations are attempted. Because of the Subsystem file type-checking, it is preferable that the calling program should not itself disconnect either FILE1 or FILE2 before EMAS3NEWGEN is called.

%externalroutinespec EMAS3OUTFILE %c
(%stringname file, %integername size, hole, prot, conad, flag) {srirririwiw}

This procedure is used to provide a new, all-zero file named FILE. SIZE is the required size (including file header) in bytes: the file provided is of this size, rounded up to a multiple of 4096 bytes. FILE may, but need not already exist. Although the file is otherwise zero, certain fields of the file header are set up as follows:

Word 0	is set equal to 32 (the default file header size and offset of start of data)
Word 1	is also set to 32.
Word 2	is set to the physical size of the file in bytes
Word 3	is set to filetype code 3 (character)
Word 5	is set with the date and time of creation (the current date and time) in the format described in the section on File Formats.

Unless HOLE is set negative before the call, the file is connected in the virtual memory in write mode, and CONAD is set to the connect address, otherwise the file is created (if necessary) but not connected, and CONAD is set zero.

HOLE provides a way of indicating the maximum expected size of FILE, after possible later calls of CHANGEFILESIZE. It is used to reserve a hole of sufficient size in the virtual memory for later extension of the file. HOLE is specified as a number of bytes, but if zero is specified a default value equal to the size of FILE rounded up to the machine's segment size is taken (1 Mbyte for IBM/XA implementations of EMAS).

EMAS3OUTFILE is the most efficient way of providing such zero files because it "forgets" any existing file contents without inducing write-outs to disc of unwanted pages, and provides zero pages in virtual memory as required without any unproductive page-in from disc.

PROT is bit significant. It should be set zero unless one of the following effects is required:

Bit value	Effect
x'00000002'	available for immediate disconnection (for example when the virtual memory hole is required for another connect)
x'00000004'	soft-permanent connection (the Subsystem will not allow disconnection of this file during the current session).
x'00000008'	hard-permanent connection (Director will not allow disconnection of this file during the current session).
x'00000040'	the parameter CONAD is set with the required connect address
x'00000080'	the left-most byte of PROT contains the number of the FSYS (file system) containing FILE
x'40000000'	a TEMP (temporary) file is required. (This can also be specified by nominating a filename whose first two characters are "T#".) A TEMP file will be destroyed at the end of the session in which it is created.
x'20000000'	a VTEMP (very temporary) file is required. A VTEMP file is destroyed when it is disconnected. If both TEMP and VTEMP are specified, then the file will be made VTEMP.

%externalroutinespec EMAS3OUTPOS (%integername pos) {iw}

POS is set equal to the current output line-length on the currently selected output stream. This information is of use only during non-record oriented i/o as used by the IMP language. After an output NEWLINE character, EMAS3OUTPOS yields zero, and thereafter the number of characters which have been output on the current line.

%externalroutinespec EMAS3PACKDATEANDTIME %c
(%stringname date, time, %integername dtword) {srsriw}

From the parameters DATE and TIME, which should be in the format, respectively, "dd/mm/yy" and "hh.mm.ss", the integer DTWORD is set with the Subsystem's binary representation of the offered date and time (described in the section on Subsystem Data Structures).

%externalroutinespec EMAS3PHEX (%integername n) {ir}

Eight hexadecimal digits representing the 32-bit value N are output on the currently selected output stream.

%externalroutinespec EMAS3PROMPT (%stringname text) {sr}

The string TEXT is output on the interactive terminal as "prompt" text requesting typed input at the terminal.

%externalroutinespec EMAS3RENAME (%stringname file1, file2, %integername flag){srsriw}

The file currently named FILE1 is renamed, to have new name FILE2.

%externalroutinespec EMAS3RETURNCODE (%integername code) {iw}

This procedure supplies the return code set by the call of EMAS3SETRETURNCODE in the previous command. [The stored return code is set to zero at the commencement of each command. Therefore, if a command does not call EMAS3SETRETURNCODE, a zero value will be obtained by programs called by the subsequent command.]

%externalroutinespec EMAS3SENDFILE %c
(%stringname file, dev, name, %integername copies, forms, flag) {srsrsrirriw}

The file named FILE belonging to the process owner is sent to output queue DEV, forms FORMS, specifying COPIES copies, attaching a symbolic identifier NAME for document queries.

%externalroutinespec EMAS3SETCOMREG (%integername i, value) {irir}

The Subsystem communication word number I is set to VALUE.

%externalroutinespec EMAS3SETFNAME (%stringname fname) {sr}

The Subsystem identifier field (frequently a filename, but also often some other text) is set to FNAME. When EMAS3SETRETURNCODE is called and the error message associated with its parameter contains an ampersand character, the FNAME text is printed in place of the ampersand. [A complete list of the Subsystem error messages for non-zero result flag values is given in the section Subsystem Error Messages and in file SUBSYS:ERRORTEXTS.]

%externalroutinespec EMAS3SETMODE (%stringname s, %integername flag) {sriw}

PAD mode information is taken from the string S and is despatched to the PAD

controlling the interactive terminal for the calling program.

%externalroutinespec EMAS3SETRETURNCODE (%integername flag) {ir}

This procedure is normally called when a Subsystem or user-written command is completed. A zero value for FLAG conventionally means that the command was "successful". Non-zero values indicate failure, or perhaps conditions which require the issue of a warning message. The Subsystem-defined values for error conditions during Subsystem command execution are given in the section Subsystem Error Messages. Where an ampersand appears in an error message the current content of the "FNAME" string most recently set by procedure EMAS3SETFNAME is substituted. Text other than those pre-defined as Subsystem messages may be printed by using flag value 100, for which the "error message" comprises a single ampersand. EMAS3SETRETURNCODE will frequently be useful for reporting to the user, where appropriate, an error during a Subsystem interface procedure (as defined in this document). The most-recently set value of FLAG is made available to the command following the current command via procedure EMAS3RETURNCODE. The stored return code is cleared to zero at the start of each command.

%externalroutinespec EMAS3SETTRAP (%integername id, class, subclass, flag) {iriririw}

A trap is set to "catch" occurrences of contingencies of class CLASS and subclass SUBCLASS. ID is an identifier obtained from a previous call of EMAS3TRAP (q.v.), which informs the Subsystem that the current procedure intends throughout some or all of its body to trap certain contingencies. Following the initial call of EMAS3TRAP, the current procedure or its enclosed subroutines may set and unset traps for one or more classes of contingency using EMAS3SETTRAP and EMAS3UNSETTRAP. If CLASS is set -1 in the call of EMAS3SETTRAP, then all classes of trappable contingency are trapped. If SUBCLASS is set -1, then all subclasses of an explicit CLASS (when CLASS is not equal to -1) are trapped. A full description of the contingency mechanism is given in User Note 65.

%externalroutinespec EMAS3SETVSSTRAP (%integerarrayname regs31, regs24) {n/a}

This procedure is for internal use by the VSS (Virtual Storage Simulator) section of the Subsystem.

%externalroutinespec EMAS3SETWORK (%integername addr, flag) {ibiw}

This procedure is used primarily by compilers in order to set up a work file of required size. It should not be used in any way which could conflict with such use by Subsystem components. A file "T#WRK" is created of size equal to the supplied value of parameter ADDR. All bytes of the file are zero and the file has the attribute "VTEMP" (very temporary): it will be destroyed on disconnection from the virtual memory. EMAS3SETWORK places the address of the start of the file created in the Subsystem communication word number 14 ("COMREG(14)"). The file should not normally be disconnected by any procedure which uses it; but if it is disconnected, the COMREG word should be set to zero.

%externalroutinespec EMAS3SIGNAL (%integername class, subclass) {irir}

This procedure causes transfer of control as though an event (contingency) of class CLASS and subclass SUBCLASS had occurred.

%externalroutinespec EMAS3STRING (%stringname qualifier, value) {srsw}

This procedure produces an string parameter value as described in User Note 62 on the parameter acquisition mechanism (PAM).

%externalroutinespec EMAS3TIME (%stringname time) {sw}

This procedure produces the current time-of-day in the form "hh.mm.ss".

%externalroutinespec EMAS3TRAP (%integername id, prot, flag) {iwiriw}

EMAS3TRAP is called to indicate that the current procedure or its enclosed subroutines may wish to trap one or more classes of contingency. The value returned in parameter ID is used in subsequent calls of EMAS3SETTRAP and EMAS3UNSETTRAP (q.v.), which determine which classes and subclasses of contingencies are to be handled. The parameter PROT determines whether, when an appropriate contingency occurs and is trapped, further contingencies are to be allowed (PROT=0) or inhibited (PROT=1). A complete description of the contingency trapping arrangements is to be found in User Note 65.

%externalroutinespec EMAS3TRIGGERTRAP (%integername class, subclass) {irir}

This procedure causes transfer of control as though an event (contingency) of class CLASS and subclass SUBCLASS had occurred, except that such transfer of control will occur only if a user trap has been set and is on scope for the relevant class and subclass. A complete description of the contingency trapping arrangements is to be found in User Note 65.

%externalroutinespec EMAS3TRIM (%stringname file, %integername flag) {sriw}

This procedure causes the physical size of FILE (that is the number of disc pages allocated to FILE) to be adjusted (if necessary) according to the data size field (Word 0) of the file header. The third word (Word 2) of the file header is set equal to the physical size in bytes (provided that the filetype word of the header (Word 3) is a valid Subsystem filetype code. The normal purpose of this routine is to trim off excess pages from the back of a file which has been made generously large to accommodate an amount of data which is unknown at the time of the file's creation. The format of a file header is described in the section on Subsystem Data Structures.

%externalroutinespec EMAS3UCSTRING (%stringname s) {sb}

All lower case letters in the parameter string S are translated into upper case.

%externalroutinespec EMAS3UCTRANSLATE (%integername addr, length) {irir}

All lower case letters in the area of virtual store of length LENGTH bytes commencing at virtual address ADDR are translated into upper case.

%externalroutinespec EMAS3UINFI (%integername entry, value) {iriw}

This procedure is provided so that programs may ascertain various characteristics of the current process, as follows:

ENTRY	VALUE
1	"FSYS" (logical disc) number on which the process owner is accredited
2	mode: foreground=1, batch=2
3	current number of interactive and batch processes on the system
4	Unused
5	current CPU limit setting in seconds
6	maximum file-size allowed in Kbyte
7	process's SYNC1 message service number
8	process's SYNC2 message service number
9	process's ASYNC message service number
10	address of a further record of process information
11	the process number for the current invocation
12	Unused
13	the invocation number of the current process (distinguishes multiple invocations of processes for the current username)
14	Unused
15	line-length of interactive terminal (ITWIDTH)
16	Unused
17	Unused
18	Unused
19	1 if resources are scarce, else 0
20	Funds left in pence
21	charge for current session in pence
22	1 if messages are inhibited, else 0
23	terminal type
24	Front-end communication stream identifier (1)
25	Front-end communication stream identifier (2)
26	Unused
27	current user stack size option setting (USTACKSIZE)
28	current init work size option setting (INITWORKSIZE)
29	Size of interactive terminal input buffer
30	Size of interactive terminal output buffer
31	Difference between actual permanent Kbyte and your limit
32	Number of free file descriptors
33	Number of free section descriptors
34	Number of free permission descriptors
35	1 if control char ints activated else 2
36	Default define size
37	255 if parities on else 127

%externalroutinespec EMAS3UINFS (%integername entry, %stringname value) {irsw}

This procedure is provided so that programs may ascertain various characteristics of the current process, as follows:

ENTRY	VALUE
1	owner name (username) for current process
2	"DELIVERY" text
3	process start time in the format "hh.mm.ss"
4	current prompt text
5	current active directory name
6	Subsystem version identifier
7	Surname of current process owner
8	Unused
9	name of holder of Funds used by current process
10	CPU type in the form "470 V/7"
11	batch job name for current process, if running in batch mode

12	Supervisor version
13	Director version
14	Interactive terminal address
15	Host system Mail address
16	Service name of host system (e.g. EMAS-A)
17	Current "USEGROUP"

```
%externalroutinespec EMAS3UNPACKDATE %c
   (%integername dtword, %stringname date)      {irsw}
```

The date from the parameter DTWORD, assumed to be in the standard Subsystem date and time format described in the section on Subsystem Data Formats, is presented in the parameter variable DATE in the form "dd/mm/yy".

```
%externalroutinespec EMAS3UNPACKTIME %c
   (%integername dtword, %stringname time)      {irsw}
```

The time from the parameter DTWORD, assumed to be in the standard Subsystem date and time format described in the section on Subsystem Data Formats, is presented in the parameter variable TIME in the form "hh.mm.ss".

```
%externalroutinespec EMAS3UNSETTRAP (%integername id, class, subclass, flag){iriririw}
```

This procedure is used to unset a trap set by a previous call of EMAS3SETTRAP (q.v.).

```
%externalroutinespec EMAS3X (%stringname command, parm, infile, outfile,
   %integername flag)                                                {srsrsrsriw}
```

This executes 'command' with parameters 'parm'. If infile is not null then any data that would have been read from the terminal is read instead from infile. Similarly, if outfile is not null then any output that would have been written to the terminal is written instead to outfile.

2. Subsystem Commands

The following is a list of the command entry points in the Subsystem BASEFILE. Each command is followed by a list of its keywords. A program invokes a command by using the procedure EMAS3, which is described in Chapter 17 of the EMAS-3 User's Guide.

C#ACCEPT	file, newname
C#ACTIVEDIR	directory
C#ALGOL	source, object, listing, errors
C#ALIAS	name, alias
C#ALIASENTRY	entry, alias
C#ALIASPROC	name, alias, directory
C#ANALYSE	file, options, out
C#ARCHIVEPERMIT	file, date, to, permission
C#ARCHIVE	file, date
C#ASSEMBLE	source, object, listing, errors
C#AWAITRESTORES	no parameters
C#BATCH	output
C#CC	source, object, listing

C#CHECKINTEGER	text, qualifier
C#CHECKLONGREAL	text, qualifier
C#CHECKSTRING	text, qualifier
C#CHERISH	file
C#CLEAR	channel
C#CLEARJOURNAL	no parameters
C#COMMANDPROMPT	text
C#CONCAT	control file
C#CONVERT	infile, outfile
C#COPY	from, to
C#COPYGROUP	group
C#CPULIMIT	minutes, seconds
C#CURRENTREFS	no parameters
C#DATASPACE	entry, file, length, offset, access
C#DEFINE	channel, file, kbyte, format
C#DEFINEMT	tapechannel, tapefilename, volume, label, recfm, blocksize
C#DELETEDOC	document
C#DELETERECOVER	file, date
C#DELETERESTORE	file, date
C#DELIVER	delivery
C#DESTROY	file
C#DESTROYGROUP	group
C#DETACH	file, seconds, control
C#DISCARD	files, dates
C#DISCONNECT	file
C#DOCUMENTS	output
C#DONATEFUNDS	amount, user
C#DUMPH	no parameters
C#EXECUTE	program
C#FILES	files, options, out
C#FORTRAN	source, object, listing, errors
C#FUNDS	no parameters
C#HAZARD	file
C#HOLDMT	volume
C#IMP	source, object, listing, errors
C#INITPARM	parm
C#INSERT	file, directory
C#LIST	files, device, copies, forms, option
C#LOADDUMP	section
C#LOADEDENTRIES	type
C#LOADEDFILES	no parameters
C#LOADPARM	parm
C#MESSAGES	messages
C#METER	no parameters
C#NEWDIRECTORY	file, pages
C#NEWGEN	newfile, existing
C#NEWGROUP	group
C#NEWPDFILE	file
C#OBEY	file, output file or device
C#OFFER	file, user
C#OPTION	keyword=string
C#OUTPUT	specific, output
C#PARM	parm
C#PASCAL	source, object,.listing
C#PASSWORD	F or B or T, or combination
C#PERMIT	file, to, permission
C#PERMITGROUP	group, to, permission
C#PRELOAD	file
C#PTRAP	no parameters
C#QUIT	no parameters

C#RECOVER	file, date, newname
C#RELEASEMT	volume
C#REMOVE	file, directory
C#REMOVEDIR	directory
C#RENAME	file, newname
C#RESETLOADER	no parameters
C#RESTORE	file, date
C#RUN	program
C#SEARCHDIR	directory, position
C#SEND	file, device
C#SETDIRMON	count
C#SETMODE	keyword=string
C#STOP	no parameters
C#SSVSN	no parameters
C#TELL	user, file
C#TERMINALTYPE	terminal type
C#TIDYDIR	directory
C#TTYLIST	no parameters
C#TIDYPDFILE	pdfile
C#USEGROUP	groupname
C#USERS	no parameters
C#WARNINGS	on or off

3. The IMP Interface

These are similar - in some cases identical - to the corresponding facilities on EMAS 2900. They are intended for IMP programmers and in most cases cannot be called from other languages.

```
%externalroutinespec accept %alias "S#ACCEPT" %c
    (%string (255) file, newname, %integername flag)
%externalroutinespec allowinterrupts %alias "S#ALLOWINTERRUPTS"
%externalroutinespec archivepermit %alias "S#ARCHIVEPERMIT" %c
    (%string(255)file, %string(8)date, %string(6)user, %integer mode,
        %integername flag)
%externalroutinespec askmag %alias "S#ASKMAG" %c
    (%integer channel, %string(7)vol, %integername flag)
%externalroutinespec assignvalue %alias "S#ASSIGNVALUE" %c
    (%stringname ref, %string(255)value, %integername flag)
%externalroutinespec bdirlist %alias "S#BDIRLIST"
%externalroutinespec castout %alias "S#CASTOUT" (%stringname line)
%externalroutinespec changeaccess %alias "S#CHANGEACCESS" %c
    (%string(255)file, %integer mode, %integername flag)
%externalroutinespec changefilesize %alias "S#CHANGEFILESIZE" %c
    (%string(255)file, %integer newsize, %integername flag)
%externalintegerfnspec charge %alias "S#CHARGE" %c
    (%longrealname cpu, %integername pageturns, connect minutes)
%externalintegerfnspec checkcommand %alias "S#CHECKCOMMAND" %c
    (%string(255)command)
%externalroutinespec checkinteger %alias "S#CHECKINTEGER" %c
    (%string(255)text, qualifier, %integername value, flag)
%externalroutinespec checklongreal %alias "S#CHECKLONGREAL" %c
    (%string(255)text, qualifier, %longrealname value, %integername flag)
%externalintegerfnspec checkname %alias "S#CHECKNAME" %c
    (%stringname name, %integername type, qualifier)
%externalroutinespec checknoparams %alias S#CHECKNOPARAMS
```

```
%externalroutinespec checkstring %alias "S#CHECKSTRING" %c
   (%string(255)text, qualifier, %stringname value, %integername flag)
%externalroutinespec choplDr %alias "S#CHOPLDR" (%stringname S, %integer i)
%externalroutinespec clear %alias "S#CLEAR" %c
   (%integer channel, %integername flag)
%externalroutinespec clearmag %alias "S#CLEARMAG"
%externalintegerfnspec close %alias "S#CLOSE" (%integer channel)
%externalroutinespec closejournal %alias "S#CLOSEJOURNAL"
%externalroutinespec closestream %alias "S#CLOSESTREAM" (%integer stream)
%externalroutinespec copy %alias "S#COPY" %c
   (%string (255) from, to, %integername flag)
%externalbyteintegermapspec combyte %alias "S#COMBYTEMAP" (%integer n)
%externalroutinespec command prompt %alias "S#COMMANDPROMPT" %c
   (%string (255)s)
%externalroutinespec compare %alias "S#COMPARE" (%string (255) one, two)
%externalroutinespec compile %alias "S#COMPILE" %c
   (%string(255)s, %string(31)entry, %integername flag)
%externalroutinespec compile2 %alias "S#COMPILE2" %c
   (%string(255)s, %string(31)entry, %integername flag, %integer loadtype)
%externalintegermapspec comreg %alias "S#COMREGMAP (%integer n)
%externalstringfnspec confile %alias "S#CONFILE" (%integer adr)
%externalroutinespec connect %alias "S#CONNECT" %c
   (%string(255)file, %integer mode, hole, prot,
      %integername conad, type, start, end, flag)
%externalroutinespec console %alias "S#CONSOLE" %c
   (%integer ep, %integername start, len)
%externalroutinespec consource %alias "S#CONSOURCE" %c
   (%string(255)file, %integername adr)
%externalroutinespec controlline %alias "S#CONTROLLINE" %c
   (%stringname line, %integername flag)
%externalroutinespec copy group %alias "S#COPYGROUP %c
   (%string (255) from, to, %integername flag)
%externallongrealfnspec cputime %alias "S#CPUTIME"
%externalintegerfnspec currentll %alias "S#CURRENTLL"
%externalintegerfnspec currentpackeddt %alias "S#CURRENTPACKEDDT"
%externalroutinespec cxdump %alias "S#CXDUMP" (%integer start, finish, type)
%externalstringfnspec date %alias "S#DATE"
%externalroutinespec define %alias "S#DEFINE" %c
   (%integer channel, %string(255)id, %integername afd, flag)
%externalroutinespec definfo %alias "S#DEFINFO" %c
   (%integer channel, %stringname file, %integername status)
%externalstringfnspec derrs %alias "S#DERRS" (%integer n)
%externalroutinespec destroy %alias "S#DESTROY" %c
   (%string(255)file, %integername flag)
%externalintegerfnspec devcode %alias "S#DEVCODE" (%string(16)device)
%externalstringfnspec devname %alias "S#DEVNAME" (%integer code)
%externalroutinespec dfdfin %alias "S#DFDFIN" %c
   (%string(255)infile, %integer channel, %integername flag)
%externalroutinespec dfdfout %alias "S#DFDFOUT" %c
   (%string(255)outfile, %integer channel, %integername flag)
%externalintegerfnspec dirtoss %alias "S#DIRTOSS" (%integer flag)
%externalroutinespec disconnect %alias "S#DISCONNECT" %c
   (%string(255)file, %integername flag)
%externalroutinespec disposeheap %alias "S#DISPOSEHEAP" (%integer adr)
%externalintegerfnspec dtword %alias "S#DTWORD" (%integer s)
%externalroutinespec dump %alias "S#DUMP" (%integer start, finish)
%externalroutinespec enable discon %alias "S#ENABLEDISCON" %c
   (%string (255) file, %integername flag)
%externalroutinespec etoi %alias "S#ETOI" (%integer addr, len)
%externalintegerfnspec eventdata %alias "S#EVENTDATA" (%integer adr)
```

```
%externalroutinespec exec %alias "S#EXEC" (%string (255) file, %integername flag)
%externalintegerfnspec exist %alias "S#EXIST" (%string(255)file)
%externalintegerfnspec existtype %alias "S#EXISTTYPE" (%string(255) file)
%externalroutinespec extend %alias "S#EXTEND" %c
    (%record(fdf)%name fd, %integername flag)
%externalstringfnspec extractvalue %alias "S#EXTRACTVALUE" %c
    (%stringname ref, %integername flag)
%externalstringfnspec failuremessage %alias "S#FAILUREMESSAGE" (%integer n)
%externalintegerfnspec fdaddr %alias "S#FDADDR"
%externalintegerfnspec fdmap %alias "S#FDMAP" (%integer channel)
%record %format arf(%string (255) name, %integer type)
%externalroutinespec fileanal %alias "S#FILEANAL" %c
    (%string(255)file, %record(arf)%arrayname r, %integername count, flag)
%externalroutinespec files %alias "S#FILES" %c
    (%string(255)pattern, %integer options, %integername flag)
%externalroutinespec fill %alias "S#FILL" (%integer len, from, filler)
%externalroutinespec fillpattn %alias "S#FILLPATTN" %c
    (%integer copies, bytes, from, to)
%externalintegerfnspec find %alias "S#FIND" %c
    (%string(31)entry, %integername rec, %integer adr, type)
%record %format frf(%integer conad, filetype, datastart, dataend,
    size, rup, eep, apf, users, arch, %string (6) tran,
        %string (8) date, time, %integer count, spare1, spare2)
%externalroutinespec finfo %alias "S#FINFO" %c
    (%string(255)file, %integer mode, %record(frf)%name r, %integername flag)
%externalstringfnspec fname %alias "S#FNAME"
%externalintegerfnspec fortrandf %alias "S#FORTRANDF" %c
    (%integer dsnum, numblocks, blksize, asvar)
%externalroutinespec fprintfl %alias "S#FPRINTFL" (%longreal x, %integer n, type)
%externallongrealfnspec fracpt %alias "S#FRACPT" (%longreal x)
%externalroutinespec fskiptmmag %alias "S#FSKIPTMMAG" %c
    (%integer channel, n, %integername flag)
%externalroutinespec fstatus %alias "S#FSTATUS" %c
    (%string(255)file, %integer act, value, %integername value)
%externalintegerfnspec f77close %alias "S#F77CLOSE" %c
    (%integer dsnum, status, %integername flag)
%externalintegerfnspec f77inquire %alias "S#F77INQUIRE" %c
    (%integer dsnum, %stringname file, %integername addr, fdt)
%externalintegerfnspec f77open %alias "S#F77OPEN" %c
    (%integer dsnum, status, access, form, blanks, recl, nrec,
    %stringname file, desc, %integername fdt)
%externalroutinespec funds %alias "S#FUNDS"
%externalroutinespec generate file %alias "S#GENERATEFILE"
    (%stringname file, %integer type)
%externalintegerfnspec getbit %alias "S#GETBIT" (%integer adr, n)
%externalintegerfnspec getheap %alias "S#GETHEAP" (%integer length)
%externalroutinespec getinteger %alias "S#GETINTEGER" %c
    (%string(255)vector, %integername value)
%externalroutinespec getlongreal %alias "S#GETLONGREAL" %c
    (%string(255)vector, %longrealname value)
%record %format optionf(%string (31) bstartfile, fstartfile, preloadfile,
    %integer arraydiag, blanklines, echo, initworksize, itinsize, itoutsize, recall, spare1,
        ustacksize,
    %longinteger initparm, %integer max, %byte %integer %array b(0:255),
    %integer define size, spare2, spare3, %byte %integer autosearch, spare4, spare5,
        spare6, charints, paritymask, default terminal type, journal pages)
%externalroutinespec get options %alias "S#GETOPTIONS" %c
    (%record (optionf) %name r)
%externalroutinespec getrestofline %alias "S#GETRESTOFLINE" (%stringname text)
```

```
%externalroutinespec getstring %alias "S#GETSTRING" %c
    (%string(255)vector, %stringname value)
%externalroutinespec groupsize %alias "S#GROUPSIZE" %c
    (%string (255) group, %integername tn, tkb, pn, pkb)
%externalroutinespec hashcommand %alias "S#HASHCOMMAND" %c
    (%string (255) command, param)
%externalroutinespec hide %alias "S#HIDE" (%integername flag)
%externalroutinespec holdvol %alias "S#HOLDVOL" %c
    (%string(6)tape, %integername flag)
%externalstringfnspec htos %alias "S#HTOS" (%integer h, places)
%externalintegerfnspec inpos %alias "S#INPOS"
%externalintegerfnspec inrec %alias "S#INREC"
%externalintegerfnspec instream %alias "S#INSTREAM"
%externalintegerfnspec int %alias "S#INT" (%longreal x)
%externalstringfnspec interrupt %alias "S#INTERRUPT"
%externalintegerfnspec intpt %alias "S#INTPT" (%longreal x)
%externalintegerfnspec iocp %alias "S#IOCP" (%integer ep, parm)
%externalintegerfnspec isatty %alias "S#ISATTY" (%integer chan)
%externalroutinespec itoe %alias "S#ITOE" (%integer addr, len)
%externalstringfnspec itos %alias "S#ITOS" (%integer i)
%externalroutinespec join %alias "S#JOIN" %c
    (%record(tabf)%arrayname tab, %integer first, last, %string(255)out,
        %integer fe, %integername flag)
%externalroutinespec joinfiles %alias "S#JOINFILES" %c
    (%string(255)filenames, %record(tabf)%arrayname tab,
        %integer mode, first, %integername last, flag)
%externalroutinespec journal off %alias "S#JOURNALOFF"
%externalroutinespec journal on %alias "S#JOURNALON"
%externalintegerfnspec kins %alias "S#KINS"
%externalroutinespec lcline %alias "S#LCLINE" (%stringname text)
%externalroutinefnspec lcstring %alias "S#LCSTRING" (%string (215) s)
%externalroutinespec lctranslate %alias "S#LCTRANSLATE" (%integer addr, length)
%externalroutinespec line %alias "S#LINE" (%stringname parms)
%externallongintegerfnspec lint %alias "S#LINT" (%longlongreal x)
%externallongintegerfnspec lintpt %alias "S#LINTPT" (%longlongreal x)
%externalstringfnspec litos %alias "S#LITOS" (%longinteger n)
%record %format ent4f(%integer code offset, gla offset, ep offset, parmword)
%externalintegerfnspec loadep %alias "S#LOADEP" %c
    (%string (31) entry, %integername type, flag,
        %record (ent4f) %name ent4, %integer locll)
%externalroutinespec loadfile %alias "S#LOADFILE" %c
    (%string(255)file, %integername flag, %integer load level)
%externalroutinespec locatemag %alias "S#LOCATEMAG" %c
    (%integer channel, type, %integername count, flag)
%externalintegerfnspec lookup %alias "S#LOOKUP" (%string (255) def,
    %stringname piece)
%externalintegerfnspec lookup array %alias "S#LOOKUPARRAY" %c
    (%stringarrayname s, %stringname piece)
%externalroutinespec lprint %alias "S#LPRINT (%longlongreal llr, %integer n)
%externalroutinespec lwrite %alias "S#LWRITE" (%longinteger val, %integer places)
%externalroutinespec magio %alias "S#MAGIO" %c
    (%integer afd, op, %integername flag)
%externalintegermapspec mapssfd %alias "S#MAPSSFD" (%integer dsnum)
%externalstringfnspec modestr %alias "S#MODESTR"
%externalroutinespec moddirectory %alias "S#MODDIRECTORY" %c
    (%integer action, %string (255) dirfile, %string (31) entry,
        %string (255) filename, %integer type, %integername flag)
%externalroutinespec modpdfile %alias "S#MODPDFILE" %c
    (%integer ep, %string(255)pdfile, %string(11)member,
        %string(255)infile, %integername flag)
```

```
%externalroutinespec monparams %alias "S#MONPARAMS" (%string(31)template)
%externalroutinespec move %alias "S#MOVE" (%integer length, from, to)
%externalroutinespec ncode %alias "S#NCODE" (%integer start, finish, addr)
%externalroutinespec ndiag %alias "S#NDIAG" (%integer pc, lnb, fault, inf)
%externalintegerfnspec newfileop %alias "S#NEWFILEOP" %c
    %integer dsnum, action, type, %integername afd)
%externalroutinespec newgen %alias "S#NEWGEN" %c
    (%string(255)file, newfile, %integername flag)
%externalintegerfnspec newmtfileop %alias "S#NEWMTFILEOP" (%integer afd, act)
%externalstringfnspec nexttemp %alias "S#NEXTTEMP"
%externalroutinespec note %alias "S#NOTE" (%string(255)s)
%record %format oldconf(%integer conad, filetype, datastart, dataend)
%externalroutinespec oldconnect %alias "S#OLDCONNECT" %c
    (%string(255)file, %integer mode, hole, prot,
        %record(oldconf)%name r, %integername flag)
%externalroutinespec offer %alias "S#OFFER" %c
    (%string (255) file, %string (6) to, %integername flag)
%externalintegerfnspec ok to overwrite %alias "S#OKTOOVERWRITE" %c
    (%integer type 1, type 2)
%externalintegerfnspec open %alias "S#OPEN" (%integer afd, mode)
%externalroutinespec openmag %alias "S#OPENMAG" %c
    (%integer channel, %string(7)s)
%externalroutinespec outfile %alias "S#OUTFILE" %c
    (%string(255)file, %integer size, hole, prot, %integername conad, flag)
%externalintegerfnspec outpos %alias "S#OUTPOS"
%externalintegerfnspec outrec %alias "S#OUTREC" (%integer length)
%externalintegerfnspec outstream %alias "S#OUTSTREAM"
%externalintegerfnspec packdateandtime %alias "S#PACKDATEANDTIME" %c
    (%string(8)date, time)
%externalintegerfnspec pageturns %alias "S#PAGETURNS"
%externalroutinespec pamhelp %alias "S#PAMHELP" %c
    (%stringname file, command, param)
%externalroutinespec pamjoin %alias "S#PAMJOIN" %c
    %string (255) files, %stringname concfile, %integername flag)
%externalintegerfnspec parmap %alias "S#PARMAP"
%externalroutinespec pdisconnect %alias "S#PDISCONNECT" %c
    (%string(255)file, %integername flag)
%externalroutinespec permit %alias "S#PERMIT" %c
    (%string(255)file, %string(6)user, %integer mode, %integername flag)
%externalroutinespec phex %alias "S#PHEX" (%integer n)
%externalroutine pop %alias "S#POP" (%stringname s, %integername flag)
%externalroutinespec print %alias "S#PRINT" (%longreal x, %integer n, m)
%externalroutinespec printchs %alias "S#PRINTCHS" (%string(255)s)
%externalroutinespec printfd %alias "S#PRINTFD" (%integer afd, mode)
%externalroutinespec printfl %alias "S#PRINTFL" (%longreal x, %integer n)
%externalroutinespec printline %alias "S#PRINTLINE" (%string (255) s)
%externalstringfnspec printparms %alias "S#PRINTPARMS" (%longinteger p)
%externalroutinespec printstringarray %alias "S#PRINTSTRINGARRAY" %c
    (%stringarrayname sa)
%externalroutinespec prompt %alias "S#PROMPT" (%string(255)s)
%externalintegerfnspec pstoi %alias "S#PSTOI" (%string(255)s)
%externalroutinespec psysmess %alias "S#PSYSMESS" (%string (255) caller,
    %integer flag)
%externalroutinespec push %alias "S#PUSH" (%string (255) s, %integername flag)
%externalroutinespec queue %alias "S#QUEUE" (%string (255) s,
    %integername flag)
%externalroutinespec read %alias "S#READ" (%name v)
%externalroutinespec readmag %alias "S#READMAG" %c
    (%integer channel, addr, %integername length, flag)
```

```
%externalroutinespec readprofile %alias "S#READPROFILE" %c
    (%string(11) key, %name data, %integername version, flag)
%externalroutinespec readstring %alias "S#READSTRING" (%stringname s)
%externalroutinespec releasevol %alias "S#RELEASEVOL" (%string (6) tape,
    %integername flag)
%externalroutinespec rename %alias "S#RENAME" %c
    (%string(255)file, newfile, %integername flag)
%externalintegerfnspec returncode %alias "S#RETURNCODE" (%integer n)
%externalroutinespec reveal %alias "S#REVEAL"
%externalroutinespec rewindmag %alias "S#REWINDMAG" (%integer channel)
%externalintegerfnspec roundup %alias "S#ROUNDUP" (%integer n, round)
%externalintegerfnspec s11ok %alias "S#S11OK" (%stringname s11)
%externalintegerfnspec samebytes %alias "S#SAMEBYTES" %c
    (%integer len, addr1, addr2)
%externalroutinespec sdisconnect %alias "S#SDISCONNECT" %c
    (%string(255)file, %integer fsys, %integername flag)
%externalstringfnspec segsinuse %alias "S#SEGSINUSE" %c
    (%integername firstseg, lastseg, %integer segstart)
%externalroutinespec sendfile %alias "S#SENDFILE" %c
    (%string(255)file, %string(16)device, %string(24)name,
        %integer copies, forms, %integername flag)
%externalintegerfnspec setbit %alias "S#SETBIT" (%integer adr, n, value)
%externalroutinespec set conlevel %alias "S#SETCONLEVEL" %c
    (%string (255) file, %integer conlevel, %integername flag)
%externalroutinespec setdefaults %alias "S#SETDEFAULTS" (%integername flag)
%externalroutinespec setfname %alias "S#SETFNAME" (%string(255)fname)
%externalroutinespec sethelp %alias "S#SETHELP" (%string (255) help file)
%externalroutinespec set io default %alias "S#SETIODEFAULT" (def, chan)
%externalroutinespec setline %alias "S#SETLINE" (%string (255) text)
%externalroutinespec setmag %alias "S#SETMAG" %c
    (%integer channel, density, xcmode, %integername flag)
%externalintegerfnspec setmode %alias "S#SETMODE" (%string(255)prm)
%externalroutinespec setpar %alias "S#SETPAR" (%string(255)s)
%externalroutinespec setreturncode %alias "S#SETRETURNCODE" (%integer n)
%externalroutinespec setssinhibit %alias "S#SETSSINHIBIT"
%externalroutinespec settopfd %alias "S#SETTOPFD" (%integer fdno)
%externalroutinespec show caller %alias "S#SHOWCALLER" (%integer, lub)
%externalintegerfnspec sizeof %alias "S#SIZEOF" (%name x)
%externalroutinespec skipmag %alias "S#SKIPMAG" (%integer channel, n)
%externalroutinespec skiptmmag %alias "S#SKIPTMMAG" (%integer channel, n)
%externalstringfnspec slprint %alias "S#SLPRINT" (%longlongreal llr,
    %integer m, n)
%externalstringfnspec slprintfl %alias "S#SLPRINTFL" (%longlongreal llr,
    %integer n)
%externalstringfnspec slwrite %alias "S#SLWRITE" %c
    (%longinteger val, %integer places)
%externalstringfnspec spar %alias "S#SPAR" (%integer n)
%externalstringfnspec sprint %alias "S#SPRINT" (%longreal x, %integer m, n)
%externalstringfnspec sprintfl %alias "S#SPRINTFL" (%longreal x, %integer n)
%externalroutinespec ssenter %alias "S#SSENTER" %c
    (%stringname strings, %integername integers, %string(31)template)
%externalroutinespec ssexit %alias "S#SSEXIT" %c
    (%stringname strings, %integername integers, %string(31)template)
%externalroutinespec sstrace %alias "S#SSTRACE" %c
    timestamp Trace: name of calling procedure
%externalstringfnspec ssversion %alias "S#SSVERSION"
%externalintegerfnspec starts %alias "S#STARTS" (%stringname line, first)
%externalintegerfnspec startswith %alias "S#STARTSWITH" %c
    (%stringname a, %string(255)b, %integer chop)
%externalroutinespec statusmag %alias "S#STATUSMAG" (%integer channel, addr)
```

```
%externalintegerfnspec stoi %alias "S#STOI" (%string (255) s, %integername i)
%externalroutinespec stop %alias "S#STOP"
%externalintegerfnspec stor %alias "S#STOR" (%string (255) s, %longrealname lr)
%externalintegerfnspec storematch %alias "S#STOREMATCH" %c
    (%integer n, adr1, adr2)
%externalroutinespec stringtolonginteger %alias "S#STRINGTOLONGINTEGER" %c
    (%string(255)s, %longintegername li, %integername flag)
%externalstringfnspec substring %alias "S#SUBSTRING" %c
    (%stringname s, %integer i, j)
%externalroutinespec tempdir %alias "S#TEMPDIR" %c
    (%string(255)file, %integername flag)
%externalroutinespec terminate %alias "S#TERMINATE"
%externalstringfnspec time %alias "S#TIME"
%externalroutinespec tojournal %alias "S#TOJOURNAL" (%integer from, length)
%externalroutinespec top n tail %alias "S#TOPNTAIL" (%stringname s)
%externalroutinespec trim %alias "S#TRIM" (%string(255)file, %integername flag)
%externalstringfnspec ucstring %alias "S#UCSTRING" (%string(255)s)
%externalroutinespec uctranslate %alias "S#UCTRANSLATE" (%integer addr, len)
%externalintegerfnspec uinfi %alias "S#UINFI" (%integer n)
%externalstringfnspec uinfs %alias "S#UINFS" (%integer n)
%externalroutinespec unload2 %alias "S#UNLOAD2" %c
    (%integer local load level, fail)
%externalroutinespec unloadmag %alias "S#UNLOADMAG" (%integer channel)
%externalintegerfnspec unok %alias "S#UNOK" (%stringname user)
%externalstringfnspec unpackdate %alias "S#UNPACKDATE" (%integer d)
%externalstringfnspec unpacktime %alias "S#UNPACKTIME" (%integer t)
%externalintegerfnspec valid header %alias "S#VALIDHEADER"
    (%string (255) file, %integer conad, %integername size)
%externalintegerfnspec vdub %alias "S#VDUB" (%integer n)
%externalstringfnspec vduc %alias "S#VDUC" (%integer y, x)
%externalintegerfnspec vdui %alias "S#VDUI" (%integer n)
%externalstringfnspec vdus %alias "S#VDUS" (%integer x)
%externalroutinespec workspace %alias "S#WORKSPACE" (%integer bytes,
    %integername start addr, flag)
%externalroutinespec wrsn %alias "S#WRSN" (%string (255) s, %integer n)
%externalroutinespec write %alias "S#WRITE" (%integer value, places)
%externalstringfnspec swrite %alias "S#SWRITE" (%integer value, places)
%externalroutinespec writemag %alias "S#WRITEMAG" %c
    (%integer channel, addr, length, %integername flag)
%externalroutinespec writeprofile %alias "S#WRITEPROFILE" %c
    (%string(11) key, %name data, %integername version, flag)
%externalroutinespec writetmmag %alias "S#WRITETMMAG" %c
    (%integer channel, %integername flag)
```

4. Subsystem Data Structures

4.1 File Header Format

The format of a file header is as follows:

Word 0 — Limit of information in the file: a count of bytes, starting from the first byte of the header; accordingly it is the byte offset in the file of the first "unused" byte in the file.

Word 1 — Byte offset of start of data in the file, equal to the size of the file header in bytes.

Word 2 Physical size of the file in bytes, equal to 4096 times the number of disc pages allocated to the file.

Word 3 File type code:
1 = object file
2 = Reserved value, used to indicate that the file does not conform to Subsystem standards.
3 = character file
4 = data file
5 = corrupt object file
6 = partitioned file
7 = directory file
8 = journal ("recall") file

Word 4 Not used

Word 5 date and time file last altered, in the format:
bit 0 equal to one
bits 1-31 the number of seconds since 00.00.00 on 1st January 1970.

Word 6 Unused except:
for Type 1 (object) files, it is the offset of the "load data" from the start of the file.
for Type 4 (data) files, when the least significant two bits indicate:
value 1 fixed-length records
value 2 variable-length records
value 3 an unstructured file ("Store mapped file")
for Type 6 (partitioned) files it is the offset of the file directory from the start of the file.
for Type 7 (directory) files, it is the offset of the start of the list of filenames/alias names.

Word 7 Unused except:
for Type 1 (object) files, it is the offset from the start of the file of the "object file map"
for Type 4 (data) files format 1 or 2, it is the number of records in the file.
for Type 6 (partitioned) files it is the number of members (used index entries).

4.2 Subsystem Date and Time Format

The Subsystem standard for storage of data and time (as an integer) is:

bit 0: one
bits 1 to 31: the number of seconds from 00.00.00 on 1st January 1970

[The sixth word, Word 5, of standard Subsystem files contains the date and time at which the file was last connected, in this format.]

Conversion between this and string (character) formats can be achieved by using the following procedure entry points (described above):

EMAS3PACKDATEANDTIME, EMAS3UNPACKDATE, EMAS3UNPACKTIME,
S#PACKDATEANDTIME, S#UNPACKDATE, S#UNPACKTIME

5 Subsystem Error Messages

These messages are also listed in file SUBSYS:ERRORTEXTS on EMAS-A.

1	Real overflow
2	Real underflow
3	Integer overflow
4	Decimal overflow
5	Zero divide
6	Array bounds exceeded
7	Capacity exceeded
8	Illegal operation
9	Address error
10	Interrupt of class
11	Unassigned variable
12	Time exceeded
13	No more space for output
14	Operator termination
15	Illegal exponentiation
16	Switch label not set
17	Corrupt dope vector
18	Illegal cycle
19	Int pt too large
20	Array inside out
21	No result
22	Param not destination
23	Arrays too large or too much recursion
24	Stream not defined
25	Input ended
26	Symbol in data
27	IOCP error
28	SUB character in data
29	Stream in use
30	Graph fault
31	Diagnostics fail
32	Resolution fault
33	Invalid margins
34	Symbol instead of string
35	String inside out
36	Wrong params provided
37	Unsatisfied reference
38	Unassigned switch variable
39	Illegal system call
40	Unrecoverable disc fault
41	Unrecoverable store or processor fault
50	SQRT arg negative
51	LOG arg negative
52	LOG arg zero
53	EXP arg out of range
54	SIN arg out of range
55	COS arg out of range
56	TAN arg out of range
57	TAN arg inappropriate
58	ASIN arg out of range
59	ACOS arg out of range
60	ATAN2 args zero
61	SINH arg out of range
62	COSH arg out of range
63	LGAMMA arg not positive
64	LGAMMA arg too large
65	GAMMA arg out of range
66	COT arg out of range
67	COT arg inappropriate
68	Real exponentiation fault
69	Complex exponentiation fault
70	RADIUS args too large
71	ARCTAN args zero
72	ARCSIN arg out of range
73	ARCCOS arg out of range
74	HYPSIN arg out of range
75	HYPCOS arg out of range
76	Matrix bound zero or negative
100	&
101	Missing left bracket
102	Missing right bracket
103	Negative sign incorrect
104	Invalid format
105	Decimal field too wide
106	Format width zero invalid
107	Repetition factor invalid
108	Null literal invalid
109	Integer field too large
110	No width field allowed
111	Literal in input format
112	Minimum digits greater than width
113	Sorry - SSMP is not available at your terminal
114	Non-repeatable edit desc
115	Comma required
116	Decimal point not allowed
117	Unit not connected
118	File already connected
119	Access conflict
120	RECL conflict
121	Form conflict
122	Status conflict
123	Invalid status
124	Form not suitable
125	Specifier not recognized
126	Specifiers inconsistent
127	Illegal specifier value
128	Invalid filename
129	No filename specified
130	Record length not specified
132	Value separator missing
133	No digits specified
134	Invalid scaling
135	Invalid logical value
136	Invalid character value
137	Value not recognized
138	Invalid repetition value
139	Illegal repetition factor

140 Invalid integer
141 Invalid real
142 Invalid subscript(s)
143 Invalid complex constant
144 Variable is not an array
145 Equals sign missing after name
146 Variable not in NAMELIST list
147 Invalid item
148 Invalid character
149 Invalid variable name
150 Literal not terminated
151 Channel & not defined
152 File does not exist
153 Input file ended
154 Wrong length record
155 Incompatible format descriptor
156 Read after write
157 Write after ENDFILE
158 Record number out of range
159 No format descriptor for data item
160 DECODE/ENCODE buffer fault
161 Invalid record size
162 No write permission for file
163 Physical end of tape
164 Invalid channel number
165 Too many files defined
166 Invalid record size
167 Invalid filename &
168 File already exists
169 Output file capacity exceeded
170 Unrecoverable system I/O error
171 Invalid operation on file
172 Wrong length record
173 No access permission
174 Invalid file description
175 File not available
176 File already open
177 Addresses inside out
178 File not open
179 File description incorrect for D.A.
180 File record size incorrect for D.A.
181 Facility not available
182 I/O error-unspecified
183 Illegal I/O operation
184 Format text too large
185 File has conflicting use
186 Blank field not permitted
187 Invalid format specification
188 RECL too large
189 NREC too large
191 F77JINIT not called
192 Delete status invalid
193 Not connected for unformatted I/O
194 Not connected for formatted I/O
195 BACKSPACE not allowed
196 Illegal BACKSPACE
197 D.A. workfiles not available
198 No filetype for a new D.A. file
199 Number of D.A. records not specified
200 Direct-access not a file property
201 Invalid username &
202 Invalid parameter &
203 CPU limit exceeds permitted maximum
204 Output limit exceeds permitted maximum
205 Resource allowance exceeded
206 Invalid control statement
207 No file access allowed
208 Data file not on-line
209 Tape access not allowed
210 Tape not available
211 CPU time exceeded
212 Job output limit exceeded
213 Termination requested
214 Invalid keyword
215 Too many parameters
216 User not registered for file use
217 Spool data area full
218 & does not exist
219 File already exists
220 Invalid filename &
221 Source library not defined
222 No object file
223 Invalid channel number
224 File name too long
225 Failed to create area
226 Corrupt object file
227 Too many external names
228 Program too large
229 File names must be different
230 File facilities not available
231 Invalid filename for level two user
232 Files in library still connected
233 &
234 File & in use at a lower level
235 File & is permanently connected
236 File & is not permanently connected
237 Read profile fails - no SS#PROFILE
238 Read profile fails - SS#PROFILE corrupt
239 Read profile fails - null key supplied
240 Read profile fails - key not found
241 Read profile fails - record in SS#PROFILE > caller's
242 Read profile fails - record in SS#PROFILE < caller's
243 Write profile fails - record length > 4060
244 Write profile fails - too many profiles
256 File & not connected
257 File & not inserted
258 Illegal use of another user's file - &
259 Illegal use of own file

260 Invalid connect mode
261 VM hole too small
262 VM full
263 Wrong number of parameters
264 Invalid device code &
265 Attempt to re-define open channel
266 Inconsistent file use
267 Invalid filetype &
268 Multiple BACKSPACE not allowed
269 Illegal use of PD file member
270 Invalid membername &
271 Attempt to write to PD member
272 Subsystem error
273 File & on offer
274 File not on offer
275 File system full
276 No free descriptors in file index
277 Too many input files
278 File connected in another VM
279 Conflicting use of file & in another process
280 User individual file limit exceeded
281 Too many permissions
282 User not in permission list
283 Own permission insufficient
284 Spooler failure
285 Looping on alias &
286 & is not a PD file
287 Member already exists
288 Member & does not exist
289 Entry & not found
290 Entry & already in directory
291 Too many entries
292 Point list full
293 Inconsistent directory entry for &
294 Illegal use of .NULL
295 Attempt to DEFINE too large file
296 Inconsistent length for data area &
297 Inconsistent parameters
298 No main entry in file
299 Attempt to load FORTRAN dynamically
300 Table too small
301 Channel not open
302 Channel is open
303 Requested access permission not allowed
304 RECALL option not selected
305 No input files
306 Duplicate request
307 Illegal call from within program
308 User total limit exceeded
309 Too many files connected
310 Attempt to overwrite PD file
311 Corrupt file &
312 Too little space for initialised stack
313 Directory & not in SEARCHDIR list
314 SEARCHDIR list full
315 Cannot OBEY within OBEY
316 Cannot call macro from program
317 Document & belongs to another user
318 Document & not queued or active
319 Attempt to write - no Write Ring requested
320 Illegal parameter format
321 Ambiguous keyword
322 Keyword not recognized
323 Too many parameters
324 Duplicated parameter
325 Missing keyword
326 Invalid value for & parameter
327 No such stream
328 No input selected
329 No output selected
330 No format information supplied in DEFINEMT
331 Spooler queue full
332 Invalid access permission
333 Group members cannot donate funds
334 Insufficient funds
335 Insufficient privilege to use device &
336 Invalid macro header
337 Too many input levels
338 Invalid parameter for READ
339 OUT parameter invalid
340 Cannot set own permission for all files
341 Cannot remove all own permissions
342 Cannot set .ALL archived files permission
343 Cannot set self permission for archived files
344 & is a filename, used where a groupname is required
345 & is a groupname, used where a filename is required
346 group & does not exist
347 Files (or groups) must be in the same file index
350 File & already loaded
351 Maximum load level exceeded
352 Chain of aliases too long
353 Entry & not loaded
354 Entry & already loaded
355 Data area not wholly within file
356 Overlaps previously defined data area &
357 & not positive integer
358 Not callable from within user program
359 Too many separate USEFOR requests

360 There is no alias inserted for &
361 Parameter mis-match
362 Cannot permit all files to everyone
363 Unsatisfied reference &
365 A null string is not acceptable
366 File name omitted &
367 Member name omitted &
368 Member name too long &
369 Invalid character in member name &
370 Name omitted
371 No digits in number &
372 Invalid character in number &
373 Number too big &
374 Length of user name & not 6
375 Invalid character in user name &
376 No file index name &
377 Invalid file index name &
378 Length of piece & > 11
379 Invalid character in piece &
380 Intermediate group name & invalid
381 Invalid date &
382 Word & not in list
383 Ambiguous word &
384 Number not in range &
385 Invalid character in option &
386 Invalid document number &
387 User & does not exist
388 Invalid device &
389 Length of user mask & not 6
390 Invalid character in user mask &
391 Group name & does not end with .
392 Attempting to & file of different type
393 Group & already exists
394 Path & already exists
395 Interactive access to tapes not permitted
396 Existing file & has different filetype
397 File & has wrong filetype
398 Cannot create all-numeric filename &
399 Inconsistent options selected
400 Number expected for parameter &
401 Unassigned variable
402 Adjustable dimension bound is unassigned
403 Assigned value is invalid
404 Assigned label is not in specified list
405 Integer is not assigned with a format label
406 Array bound exceeded
407 Array parameter upper bound is less than lower bound
408 Array parameter declared size is greater than actual
409 Assumed size array requires zero last dimension
410 Character array param only valid for FORT77 call
411 Invalid character substring position value
412 Character param declared size is greater than actual
413 Inconsistent permission &
415 Do loop increment is zero
418 Recursive call to a procedure
419 Wrong type or size of function
421 Wrong number of parameters
422 Wrong type or size of parameter
423 File & is marked for archiving
424 Negative unit number specified

425 Fault no.
426 A contained file is currently in use
427 Max depth of nested Pdmembers
428 Filename already in directory
429 Max file definitions for single filename
430 Load fail ("Maxcommon")
(Should not occur)
431 Load fail ("Duplicate area")
(Should not occur)
432 Load fail ("Area absent")
(Should not occur)
433 Monitor file already set
434 No free list cells on heap
435 No hole big enough on heap
436 No data on heap here to dispose of
437 No heap with this ID
452 Variable name longer than 11
454 Too many variables
455 Empty expression
456 Unrecognised character
457 Invalid string resolution construct
458 * is not a string operator
459 / is not a string operator
460 Unassigned variable
461 No right hand side in boolean expr
462 Boolean operator invalid
463 String resolution fails
464 Name & not known
465 Missing operator
466 Invalid statement
467 This function needs parameters
468 Function name must be followed by '('
469 Bad syntax in parameter list
470 No closing quote in string
471 Consecutive operators
472 Not a unary operator
473 Missing left bracket
474 Could not evaluate expression
475 Fault no.