UNIVERSITY OF EDINBURGH

EMAS 2900:

SUBSYSTEM WRITER'S MANUAL

A guide to the development of a subsystem for the multi-access operating system EMAS 2900

by J.K. Yarwood

edited by J.M. Murison

Preliminary edition: June 1979

EMAS 2900:

SUBSYSTEM WRITER'S MANUAL

A guide to the development of a subsystem for the multi-access operating system EMAS 2900

By J.K. Yarwood

Edited by J.M. Murison

June 1979

CONTENTS

CHAPTER 1
INTRODUCTION

Programs running in a user process in EMAS 2900 have access to

* the un-privileged instruction set of the 2900 series

* a virtual memory, currently of 32 Mbytes

* a set of procedures, collectively known as the "Director interface", which enables the process to do the following:

    a) create and access files

    b) perform interactive I/O

    c) handle error and asynchronous contingencies

    d) access "private" magnetic tapes

    e) send and accept inter-process messages

  The Director interface is described in detail in Appendix 1.

Since the procedures comprising the Director interface do not provide facilities for program compilation, loading or execution, it is assumed that most users of the System will wish either to adopt an existing set of such facilities or to write their own. Such a set of facilities is known as a "subsystem". The appearance of EMAS 2900 to a user is determined by the appearance of the subsystem he is using.

There is available with EMAS 2900 the "Edinburgh Subsystem", which provides facilities for developing and running user programs and packages. This subsystem is used in Edinburgh University; its appearance to users is described in Ref. 1.

It is not intended, however, that the Edinburgh Subsystem should provide the only user interface to the System, but that other installations should be free to provide their own subsystems, reflecting their particular requirements. This manual describes the facilities available to the writers of such subsystems.

Familiarity with "EMAS 2900: Concepts" (Ref. 2) is essential.

It has been a design principle of EMAS 2900 that a user process may not interfere, intentionally or otherwise, with any other process. At the same time, if a subsystem does not correctly use the Director interface, the process can easily disappear without trace, particularly before establishing workable contingency handling. The facilities described here are intended to allow a subsystem to be built in an incremental way from an interactive terminal, and avoiding such pitfalls.

Since a very considerable amount of basic work has already been established in the form of the Edinburgh Subsystem, it would be reasonable to expect that:

a) A new subsystem would be prepared using the facilities of the Edinburgh Subsystem.

b) A new subsystem would be created in the Edinburgh standard object file format.

However, the extent to which Edinburgh Subsystem conventions have been assumed at the Director interface is minimal, and is explained where appropriate in this manual.

CHAPTER 2
TERMINOLOGY AND CONVENTIONS

In general, use should be made of the "EMAS 2900: Concepts" manual (Ref. 2), which contains a glossary of terms used in this and other EMAS 2900 documents. The following notes supplement that glossary and describe additional relevant terms.

Local Controller, Supervisor

In this manual the terms "Local Controller" and "Supervisor" are used to describe resident supervisory code concerned with controlling the user process.

File system number, disc number

EMAS disc-packs have two decimal digits as the last two characters of the 6-character label, e.g. EMAS02. The decimal number which the pair of digits represent is called the disc number or the file system number (interchangeably).

System disc

System start-up comprises IPL followed by the loading into main store of a Supervisor (Global Controller, Local Controller and device handlers) from a specified or default site on a specified disc. That disc is called the "System disc" for the session, since from that same disc various System components are used by default: Director, permanent processes 1, 2 and 3 (respectively called DIRECT, VOLUMS and SPOOLR), and the Edinburgh Subsystem. The sites containing the code and GLAP (see below) for these components are connected into the virtual memory of each user process, as appropriate, and paged from their respective sites on the System disc.

GLAP, GLA

In the Edinburgh standard object file format, code and certain other areas are shareable. However, the area which contains the following components is fundamentally unshareable; it also needs to be written to at program load time and during execution:

a) global variables (perhaps having initialisation values)

b) references to code and data areas in other modules

c) code and data entry descriptors in this module

This area in the object file is called the GLAP (General Linkage Area Pattern). The initial action of a program load is to connect the object file into virtual memory, and then to make a copy of the GLAP into another segment which is writeable, and unshared by any other user process, leaving the complete object file (including the GLAP) available for shared use by other processes. The copy of the GLAP in the unshared segment is called the GLA (General Linkage Area).

Main log

System components are able to note chosen events in a System file called a "main log". When a current main log file is full, or when the command D/NEWLOGFILE to process no. 1 (DIRECT) is given, a new file is supplied and the old one is passed to process no. 3 (SPOOLR). It is queued by SPOOLR until the JOURNAL system (Ref. 3) accepts it for analysis and long-term storage. When the D/NEWLOGFILE command is given, a copy of the current file is also sent to the line printer queue, in SPOOLR.

Director is able to monitor process start-up and other events - as described in this manual - by writing either to the main log or, if specified, to a file belonging to the process owner.

IMP programming language

In this manual, programming entities and record formats are described in terms of the IMP programming language (Ref. 4).

CHAPTER 3
PROCESS START-UP AND ENTRY TO A SUBSYSTEM

3.0 Start-up and entry to Director

Processes are created by communication with the permanent process no. 1, DIRECT, in three ways:

* By the arrival of a log-on message from the front-end network.

* By a (batch) start-up message from SPOOLR.

* By an operator command to DIRECT: D/START username

DIRECT creates three work files to be used by the new process, respectively for the process stack, the Local Controller stack and the Director GLA (see Chapter 2). The disc addresses of the three work files are passed to the Supervisor as a "start process" request.

If the start-up is successful, DIRECT sends a message to the new process with further information, such as the reason for start-up and the interactive terminal number. The new process also receives information direct from the Supervisor as it starts. The Supervisor has connected, in the virtual memory of the new process, the Director code segment (held in one of four fixed positions on the System disc) as segment 2, the Director GLA file as segment 3, and the process stack as segment 4.

Entry to the Director code is at the byte offset in segment 2 contained in the second word of the Director code file. Director first makes a copy, into segment 3, of its own GLAP from the (back of the) code file, and then analyses the "reason for start-up" and other information. It then prepares to enter the subsystem.

Director prints out on the main log (see Chapter 2) a report of the process start-up, and then searches for the process owner's file index. The disc number containing the index may be known (e.g. in the case of batch start-up); otherwise all on-line discs are searched, starting with the System disc.

3.1 Fields in the file index

The file index contains several fields which specify how the process should run. Some of these fields are determined exclusively by the System Manager; others are controlled by the process owner if he is running at ACR level 5 or less. These fields can be set by use of the Director procedure DSFI, invoked either from the System Manager's process MANAGR or in a previous invocation of the user's process. Some of the fields may also be set by commands to the DIRECT process.

The following is a list of fields relevant to process start-up:

Field	DSFI TYPE no.	DIRECT command
Basefile name	0	D/SETBASEF D/BASEF
Interactive and batch passwords	5	
ACR level at which the process is to run	7 (priv.)	D/ACR
Director version number	8 (priv.)	
Process stack file size	10	
Process concurrency limits	14 (priv.)	
Director monitor level	16	
Contingency monitor level	17	
Director monitor file name	19	
Test basefile name	35	

The items marked "priv." may be set only by processes running at an ACR level less than the default subsystem ACR level (currently 6), and in particular by the MANAGR process. ACR 5 currently gives access to all Director "privileged" facilities.

The passwords and concurrency limits items are used by DIRECT before process start-up; the rest are relevant as described below.

3.2 Work files

First, a further work file, #SIGSTK, is created by the new process to form its contingency (or "Signal") stack. This file is connected into the virtual memory of the new process, currently as segment 6. When a contingency occurs, the Supervisor enters (or re-enters) a second process, using the segment as process stack; this second process is a "co-process" of the user process. The code executed in the co-process is the Director routine SIGNAL, which implements the contingency arrangements defined as part of the Director interface (Appendix 1).

Next, the basefile (see Section 3.1) is connected into the virtual memory as segment 32, and 33 if necessary. This is the object file containing the code and GLAP of the subsystem. If the basefile identifier is null then the default subsystem is used: this is assigned a 512 Kbytes fixed site on the System disc, currently starting at epage number X380. Otherwise the file specified must exist and the process being started must have read and execute access to it. If the basefile is unavailable for any reason, the default subsystem is entered, a message of the following form having first been sent to the main Oper console:

```
BASEFILE CONN FAIL n
```

where n is an error code specified in the Director interface definition (see the end of Appendix 1). The basefile name is held in the process file index (see Section 3.1). It must be of the form

```
        <filename>
   or   <username>.<filename>
```

where <username> has 6 characters and <filename> has up to 11 characters.

3.3 Environmental information for the subsystem

Next, Director creates and connects two more work files, a "basegla", #BGLA, and #UINFI, to contain details about the process for the subsystem.

#BGLA is to be used by the subsystem to contain its GLA (a copy of the GLAP from the basefile). It is connected at the next available segment after the subsystem code segment.

#UINFI contains data in the following format:

```
recordformat UINFF (string(6) USER, string(31) BATCH CMD FILE, c
   integer MARK, FSYS, PROCNO, ISUFF, REASON, BATCH ID, SESS IC LIM, c
   SC IDENS AD, SC IDENS, STARTCNSL, AIOSTAT, SCT DATE, SPARE1, c
   SPARE2, SPARE3, AACCT REC, AIC REVS, string (15) JOBNAME, c
   string(31) BASEFILE NAME)
```

Fields in this segment of potential use to a subsystem are as follows:

USER	The six-character name of the process owner.
BATCH CMD FILE	For a batch process (see REASON) the name of a file containing commands to be executed.
MARK	Currently 1; to be used in achieving forward compatibility if this record format is to be drastically changed.
FSYS	The file system (disc number) on which the user's file index and files reside.
PROCNO	The process number.
ISUFF	A single digit, held as an ISO character, which serves as an "invocation number" for the process. If more processes than one belonging to a single user are to be concurrent, a subsystem may wish to create work files having distinct identifiers by concatenating this character to a standard file name.
REASON	This gives the reason for the creation of the process: 0 = interactive log-on 1 = started from an OPER console 2 = started as a batch (non-interactive) process by SPOOLR.

BATCH ID	A unique integer identifier associated with the batch job (for REASON=2).
SESS IC LIM	The maximum number of thousands of machine instructions which may be executed during the current session by the process. For interactive sessions this is currently a very large number. For batch sessions it is a number given when a job is submitted to the SPOOLR process for queuing and subsequent execution. If this limit is reached by the process, it is terminated with a message at the main Oper console: SIGNAL FAIL 4 A subsystem should preferably guard against this occurrence by keeping an eye on the number of instructions executed in the current session, using Director procedure DSFI (TYPE=21).
SC IDENS AD, SC IDENS	These are used to acquire access to the Director procedures, as described in Chapter 4.
STARTCNSL	Gives the logical number of the Oper console from which the process was started (for REASON=1).
AIOSTAT	The address of a record describing interactive I/O status. The format of the record is: (<u>integer</u> IAD, <u>string</u>(15) INTMESS, <u>c</u> <u>integer</u> INBUFLEN, OUTBUFLEN, INSTREAM, OUTSTREAM)
SCT DATE	This may be used in conjunction with SC IDENS AD and SC IDENS.
JOBNAME	A character string identifier associated with the batch job when it was submitted to the SPOOLR process (for REASON=2).
BASEFILE NAME	The name of the basefile in use by the process.

3.4 Entry to the subsystem

Finally, Director connects the System Call Table and procedure identifiers (described in Chapter 4), and then enters (i.e. calls) the subsystem. This it does as follows:

* The processor PC is set to point to the address which is the basefile segment address plus the contents of the second word of the basefile.

* The processor LNB is set to the start of the processor stack segment, currently local segment 4.

* The processor SF is set to be 7 words greater than LNB.

* The processor PSR is set to contain the ACR level equal to that specified in the ACR field of the index (see Director procedure DSFI, TYPE=7; a zero field in the index specifies the default subsystem ACR level, currently 6). Settings of other PSR bits are not specified.

The subsystem is then entered as though by standard 2900 procedure CALL with two 32-bit (integer) parameters. The two parameters are:

1) An integer = 1, to be used to achive compatibility in the event of future changes.

2) The address of the UINFF record, described in Section 3.3 above.

The first action of the subsystem should be to copy the GLAP, which follows the code in the basefile, into the GLA segment. Apart from the copying, the code which does this may only access the code segment and the stack segment.

CHAPTER 4
ACCESS TO DIRECTOR PROCEDURES

This chapter describes how a subsystem acquires access to the Director procedures by forming System Call descriptors from the pairs of values (i,j) associated with the identifiers of the procedures forming the Director interface. Knowledge of the 2900 Series System Call mechanism (Ref. 5) is assumed.

The fields SC IDENS AD, SC IDENS and SCT DATE in the record format UINFF, described in Section 3.3 above, are relevant.

SC IDENS AD is the address of a record array whose elements have the following format:

```
   recordformat SC IDF (string(31) IDEN, integer I, J)
```

SC IDENS is the number of elements in the record array.

In the Edinburgh standard object module format, calls to external routines in modules compiled separately involve a machine instruction CALL with a 64-bit operand which is a descriptor descriptor. If the operand instead is a System Call descriptor then the System Call mechanism is invoked, providing controlled access to higher-privilege procedures (such as the Director procedures).

When a version of Director is created, a list of identifiers specifying permitted procedure entry points is referenced, a new System Call Table entry is created, and the identifier and (i,j) values are placed in the record array described above. The task of a subsystem requiring access to a Director procedure is thus that of selecting the correct (i,j) pair from the record array, creating a System Call descriptor containing the (i,j) values, and making that descriptor the operand of a CALL instruction.

Subsystem writers using the Edinburgh standard object module format should consult Ref. 6.

The field SCT DATE has been provided to remove the need for a subsystem to satisfy its references to Director procedures dynamically, and to enable them to be fixed up when the subsystem is created. SCT DATE is a unique identifier associated with a given version of Director. If the SCT DATE current at the time a subsystem is entered is identical to that current when the subsystem was created, then clearly the (i,j) values and entry identifiers obtaining when the subsystem was created will serve for invocations of the new subsystem. The new subsystem will of course have to retain code for fixing up Director references dynamically should it find that SCT DATE is different from that current when it was created.

CHAPTER 5
DIRECTOR MONITORING FACILITIES

5.0 Testing a new subsystem

This section addressses the practicalities of testing a new subsystem. In particular it draws attention to the difficulties of testing code which initially has no access to the terminal I/O procedures and which cannot satisfactorily diagnose programming errors (that is, until correct operation of the contingency mechanisms has been mastered).

We first describe in more detail the fields in the file index (see Section 3.1) which are specially relevant to this situation, namely:

Field	DSFI TYPE value
Base file name	0
Director version number	8
Director monitor level	16
Contingency monitor level	17
Director monitor filename	19
Test basefile name	35

Director version number, Director monitor level and contingency monitor level are cleared to default values following a System IPL.

5.1 Test basefile name (DSFI, TYPE=35)

We have stated that "basefile name" specifies the name of the object file that is to be entered as the "subsystem", and that the version residing on a fixed site on the System disc is used if that name is null. For a process belonging to the file index owner, For a process belonging to the file index owner, "test basefile name", if non-null, overrides the setting of "basefile name", even if the latter is null, for the next invocation only. Thus if the subsystem under test fails disastrously the next log-on will yield the default or specified subsystem, which can then be used to study the failure and remake the test subsystem.

5.2 Director monitor filename (DSFI, TYPE=19)

The Director monitor filename, if non-null, specifies a file belonging to the file index owner, into which Director places the following:

* Process failure messages not diagnosed by the subsystem.

* Extra monitoring of Director procedure entries (see Section 5.5: Director monitor level), or extra monitoring of contingencies (see Section 5.4: Contingency monitor level).

* Text specified, by the process, to Director procedure DMONITORTEXT.

So that a clear picture of what has happened to a new subsystem can be obtained, this file should preferably be new (all zeroes except that the third word must contain the file size in bytess) and be at least 16 Kbytes in size. The format of the file, after Director has placed text in it, is as follows. The first 32 bytess have format:

```
recordformat F (integer NEXT FREE BYTE, TEXT REL START, MAXBYTES, c
                        ZERO, SPARE1, SPARE2, NEXT CYCLIC, SPARE3)
```

As stated, MAXBYTES must first have been set to the file size in bytess.

TEXT REL START, NEXT FREE BYTE and NEXT CYCLIC are pointers relative to the start of the file. They specify the area of the file into which text has been placed, as follows:

* If NEXT FREE BYTE has been set by Director equal to MAXBYTES, then the area between TEXT REL START and MAXBYTES is being used as a circular text file, and NEXT CYCLIC specifies the bytes into which the next item of text will be or would have been written.

* If NEXT FREE BYTE is less than MAXBYTES then the file has not (yet) been filled up by Director. In this case NEXT CYCLIC is maintained equal to NEXT FREE BYTE.

If the file is not available to Director for any reason, or if the pointers are not satisfactory, no text will be placed in the file. If MAXBYTES is set greater than the actual length of the file, the process is likely to terminate in disorder.

With regard to "test basefile name" (DSFI TYPE=35), when Director has connected the monitor file at process start-up it sets the file index field holding the monitor filename to null so that the next invocation of a process belonging to the file index owner will be able to inspect the Director monitor file of the test session which failed.

Likely contents of the Director monitor file for a new subsystem might indicate:

* Process failed with no contingency information set ("SIGNAL FAIL 1")

* Process failed with repeated contingencies ("SIGNAL FAIL 3"), when attempted diagnosis of a failure also fails.

5.3 Primitive subsystem monitoring

It would be prudent for a new subsystem to place diagnostic messages itself into the file, by calling the Director procedure DMONITOR TEXT, for conditions such as being unable to set up the interactive terminal I/O (indicated by a non-zero result from Director procedure DENABLE TERMINAL STREAM).

5.4 Contingency monitor level

For the case of being unable to satisfy correctly references to the Director procedures, and for other programming errors not otherwise diagnosable, the file index field "contingency monitor level" will be useful. This field is zeroed at System start-up; it can be set by a call of Director procedure DSFI (TYPE=17). Bits in this word have meanings as follows:

Bit value	Meaning and use
2**0	Print a routine trace-back on the occurrence of a contingency, with values of variables, etc. This bit should not be used unless the subsystem is being programmed in the IMP language, since failure in Director diagnostics can lead to the process stopping in disorder.
2**1	Print text describing the contingency in words and codes ("CLASS/SUBCLASS").
2**2	Print a hexadecimal dump of the stack segment at the time of the contingency.
2**3	Print a hexadecimal dump of Director GLA at the time of the contingency (not useful to subsystem writers).
2**4	Print the contents of the virtual memory (relating segment number to filename and to disc address) at the time of the contingency.
2**5	Print a de-assembly from the code segment of the area around the value of the PC register at the time of the contingency.
2**6	Print out the machine registers at the time of the contingency.

A minimum useful value for this field is perhaps 66. Values 4 and 32 will be invaluable, and 16 may deliver useful assistance when a subsystem is becoming established.

In order to diagnose rapidly the situation in which a subsystem has been unable to satisfy its references to Director procedures, it is useful to have a recognisable pattern in the places where the System Call descriptors are intended to be placed. With the contingency monitor level 64, that pattern will be found in the printing of the machine registers DR0 and DR1 if the filling of the System Call descriptor was unsuccessful.

The contingency monitor level field in the file index can also be set by typing, at the Oper console, a command of the form:

```
        n/SIGMON m 0
```

where	n	is the process number (as displayed on the Oper screen)
	m	is an integer specifying the required value of the field, as above

5.5 Director monitor level

The Director monitor level fields in the file index comprise a bitmask (32 bits) specifying which Director procedures are to be monitored at entry, and an integer DEPTH, specifying the depth of monitoring to be supplied. Not all Director entries can be monitored using this mechanism, partly for reasons of efficiency in a working system, but the following bits have been allocated:

Bit value	Director procedure
2**1	DCREATE
2**2	DDESTROY
2**3	DCONNECT
2**4	DDISCONNECT
2**5	DFILENAMES
2**6	DRENAME
2**7	DFINFO
2**8	PRIME CONTINGENCY
2**9	READID
2**10	DISCID
2**11	DPERMISSION
2**12	DOFFER
2**13	DACCEPT
2**14	DSETIC
2**15	DFSTATUS
2**16	DNEWGEN
2**17	DCHSIZE
2**18	DTRANSFER
2**19	DSTOP
2**20	DSFI
2**21	DNEW OUTWARD CALL
2**22	DNEW INWARD CALL
2**23	DNOMINATE STACK
2**24	ACREATE

Bit value	Director procedure
2**25	ADESTROY
2**26	DRESTORE
2**27	DMOD ARCH
2**28	DMESSAGE

The form of the monitoring is that of the IMP language run-time diagnostics. In addition, at procedure exit the result of the Director procedure is printed in the form R=n. n is -1 if a result is not relevant.

If DEPTH is zero, the Director procedure name is printed and, provided that a non-optimised version of Director is being used, the values of the parameters to the procedure are also printed. DEPTH should be zero unless the subsystem is being written in a language which generates Edinburgh Subsystem standard diagnostics; a non-zero value specifies that an extra trace-back of procedures calling the Director procedure is to be printed.

5.6 Director version

A System disc has four fixed sites (256 Kbytes each), for up to four versions of Director. Site zero is used by default, and will normally contain a Director compiled without symbol tables for run-time diagnostics. In this case the monitoring described in Section 4.5 above will be of minimal value, as only the procedure name will be printed and the values of parameters will not appear. It may be possible to assign a non-default Director site to contain a version compiled to include symbol tables. Setting the Director version field in the index causes the next interactive log-on by the process owner to use the specified Director version (0, 1, 2 or 3) rather than the default version (0). This facility should only be used after consultation with the System Manager, as the contents of a given Director site cannot always be guaranteed to be constant, or even valid.

CHAPTER 6
PROCESS TERMINATION

When a process terminates under control of Director (as should always be the case in principle), Director prints a message, both on the main log and in the process's Director monitor file if applicable, indicating the reason for stopping. If the subsystem requested termination of the process by calling Director procedure DSTOP, the integer parameter REASON is printed. However, if Director initiates the stop sequence, for example because a contingency has been incorrectly handled, then a REASON is printed from the list given below.

Since these values are pre-assigned, a suggested convention is that subsystems use REASON=100 for a normal stop, and values 101-199 to specify detected abnormal conditions.

REASON	Meaning
0	Error condition noted in monitor printing (main log and/or monitor file).
1	A contingency occurred, but the Director procedure PRIME CONTINGENCY (used to specify a subsystem contingency procedure to be executed) had not previously been called.
2	A program error has occurred during execution of the subsystem's contingency handling procedure and before a call of Director procedure DRESUME. (A call of DRESUME indicates that diagnostic actions are complete, and more specifically that the contingency procedure itself is again ready to handle further contingencies.)
3	The number of program error and virtual store contingencies for the process has exceeded a certain fixed number, currently 32. The purpose of this limit is to terminate the contingency loop which will occur if the subsystem contingency procedure executes satisfactorily but the computation repeatedly "resumed to" immediately fails.
4	The number of instructions executed by the process exceeds the limit specified for the session. In the case of an interactive session this is currently a very large number, but may be subject to the System Manager's control. In the case of a batch session, it is the number specified when the batch job was submitted to the SPOOLR process. A subsystem should normally arrange, through use of Director procedures DSETIC and DSFI (TYPE=21), that this session limit is not violated, in order to initiate the termination under its own control.
5	Not used.
6	A program error has occurred during execution of a Director procedure. Currently this may be caused by supplying the wrong number of parameter words to a procedure; later machine modification levels will enable this condition to generate a "subsystem program error" contingency.

7 Illegal call of Director procedure DRESUME: the value of the LNB parameter is >0 but is not at least 5 words below the machine register LNB contents at the time of the call of DRESUME.

8 Illegal call of Director procedure DRESUME: the parameter LNB specifies Director's contingency stack segment, which is reserved.

9 Illegal call of Director procedure DRESUME, specifying resumption of a computation in which a virtual store error (address error) has just occurred.

10 Illegal call of Director procedure DRESUME, specifying resumption of a computation when a contingency has not in fact occurred.

11 A processor stack-switch has failed to occur, perhaps when a call of Director procedure DRESUME has specified (through parameter LNB) a segment which is not the normal or other nominated stack segment.

12 Illegal call of Director procedure DASYNC INH to despatch (accept) a queued asynchronous contingency, either when no contingency is queued, or before the subsystem contingency-handling procedure has indicated, through a suitable call of Director procedure DRESUME, that it is able to accept further contingency notifications.

13 An "emergency stop" command has been sent to the process Director from the machine Operator console. This REASON for stopping may be printed in addition to one of the above-specified stopping messages when certain of the above failures occur during processing using the subsystem-nominated processor stack segment. In this case the message printed previously specifies the true reason for stopping.

14 A "stop" command has been sent from the machine Operator console or from the permanent process no. 1 (DIRECT) as part of the System automatic close-down sequence, or from the interactive communications system following line or network-processor failure.

If a process is terminated by the Supervisor (specifically the Local Controller) because a program or virtual store error has occurred during execution of the Director contingency-handling code, the only evidence of process termination will be on the main log, in the form

```
PROC:n keyword 6/m
```

where n is the process number, m is a page number in segment 6 and "keyword" specifies the contingency type; for example, VSERROR, INSTRUCTION, DESCRIPTOR, STACK. Process termination in this manner is exceptional and disastrous as regards restarting a process for the file index owner, because files belonging to the owner will remain marked as connected in some virtual memory. This condition can be cleared only by reloading the System or by running a special procedure under the System Manager's supervision.

References

1 EMAS 2900: User's Guide (describes the facilities of the Edinburgh Subsystem). ERCC (1979).

2 EMAS 2900: Concepts. ERCC, 2nd Edition (1978).

3 EMAS 2900: The JOURNAL System. ERCC (1979).

4 The IMP Language and Compiler. Stephens, P.D., Computer Journal, Vol. 17 no. 3 (1974).

5 The Primitive Level Interface. ICL internal document PSD 2.5.1 (1975).

6 EMAS 2900 Subsystem Note 9: Edinburgh Standard Object File Format. ERCC (1979).

7 The EMAS Director. Rees, D.J., Computer Journal, Vol. 18 no. 2 (1975).

8 EMAS 2900 System Note 4: EMAS 2900 System Calls. ERCC (1977).

9 EMAS 2900 Supervisor Note 4: Local Stacks. ERCC (1977).

10 EMAS 2900 Supervisor Note 1: PON Mechanism for Director. ERCC (1978).

APPENDIX 1

THE DIRECTOR INTERFACE

Contents

A1.1 Introduction

As explained in the introduction to the main text of this document, the facilities available to a subsystem in EMAS 2900 are provided by a set of procedures collectively known as the "Director interface". This appendix describes each of these procedures in detail.

Section A1.2 comprises an alphabetical list of the Director interface procedures, with a summary of their functions and page number references to their detailed descriptions.

Section A1.3 describes the EMAS 2900 file system, the creation and maintenance of which is one of Director's tasks. The relevant procedures are then described in Section A1.4.

Each of the remaining sections likewise describes a logical group of Director procedures, apart from the final section, A1.10, which comprises a list of all the error messages used by the procedures.

A1.1.1 General notes

These apply where explicitly indicated in this appendix.

Note 1. Except for privileged processes, USER must be the process owner.

Note 2. Privileged processes may own indexes on several disc-packs. It is only necessary to specify FSYS for such situations; otherwise you may set FSYS to -1. However, a subsystem may know (for example, from a call of procedure DUSERNAME) on which disc-pack USER resides, and specifying that disc-pack as FSYS may speed the search time for the file index for USER.

Note 3. If the file referred to is on archive storage, the TYPE parameter is used to indicate this, and the DATE parameter must specify the date of archiving in the form "dd/mm/yy", where dd, mm and yy represent day, month and year respectively. The DATE parameter is ignored if the TYPE parameter indicates that the file is in disc storage rather than archive storage.

In addition, the reader is referred to Chapter 2 of the main text, which explains some of the terminology and conventions used in EMAS 2900 documentation, in particular in the procedure descriptions below.

A1.2 Procedures in the Director Interface

Procedure	Page	Description
ACREATE	A1-23	Creates an entry for a file in an archive index.
DACCEPT	A1-11	Accepts a file offered to the caller by another user.
DASYNC INH	A1-35	Causes asynchronous contingencies to be inhibited.
DCHSIZE	A1-10	Changes the size of a file.
DCLEAR INT MESSAGE	A1-30	Clears a multi-character terminal INT: message.
DCONNECT	A1-9	Connects a file into the caller's virtual memory.
DCREATE	A1-8	Creates a file.
DDESTROY	A1-8	Destroys an on-line or archive file.
DDISABLE TERMINAL STREAM	A1-29	Suspends or aborts an I/O stream established using DENABLE TERMINAL STREAM.
DDISCONNECT	A1-20	Disconnects a file from the caller's virtual memory.
DENABLE TERMINAL STREAM	A1-26	Sets up the connection between a stream and a file to be used for terminal input or output.
DFILENAMES	A1-20	Gives details of all a user's disc files or archive files.
DFINFO	A1-14	Gives information about a file.
DFSTATUS	A1-15	Modifies the attributes of a file.
DFSYS	A1-19	Gives the number of the disc-pack on which a user's files reside.
DGETDA	A1-21	Gives the disc addresses of the sections of a file.
DISC ID	A1-34	Discards interrupt data and uninhibits asynchronous contingencies.
DMAG CLAIM	A1-42	Claims or releases a magnetic tape.
DMAG IO	A1-42	Carries out an operation with respect to a previously claimed magnetic tape.

DMESSAGE	A1-41	Sends or receives a messages from or to an address, or determines whether a user currently has a process.
DMOD ARCH	A1-24	Modifies entries in an archive index.
DNEW ARCH INDEX	A1-23	Creates an archive index for a user.
DNEWGEN	A1-20	Enables a new version of a file to be introduced while the current version is connected.
DNEW INWARD CALL	A1-37	Creates a System Call Table entry for an inward call.
DNEW OUTWARD CALL	A1-36	Creates a System Call Table entry for an outward call.
DNOMINATE STACK	A1-36	Specifies to the caller's Local Controller which segment is to be SSN+1.
DOFFER	A1-11	Offers a file to a user.
DOUT	A1-39	Sends a System message and causes the process to suspend pending a reply.
DOUT11	A1-40	Sends a System message and keeps the process's pages in main store pending a reply.
DOUT18	A1-40	Sends a System message with various special effects.
DPERMISSION	A1-12	Sets access permissions or gets details of access permissions with respect to a file or a complete file index.
DPOFF	A1-39	Causes the process to suspend pending the receipt of a message.
DPON	A1-39	Sends a System message and allows the process to continue.
DPON2	A1-38	Sends a System message.
DPRG	A1-22	Moves the contents of a file to a disc site.
DRENAME	A1-11	Renames a file belonging to the caller.
DREQUEST TERMINAL OPERATION	A1-26	Requests a terminal output or input message.
DRESET CONTINGENCY	A1-35	Causes the first routine nominated via PRIME CONTINGENCY to be used again henceforth.
DRESTORE	A1-23	Passes a restore request, in respect of an archive file, to VOLUMS.
DRESUME	A1-33	Informs Director of various stages in the execution of the routine nominated via PRIME CONTINGENCY.

DSFI	A1-17	Reads or sets items in the SFI (System File Information) of a file index.
DSPOOL	A1-41	Generates a spool request to SPOOLR in respect of a file.
DTOFF	A1-39	Seeks a System message but does not cause the process to suspend if there is none.
DTRANSFER	A1-21	Transfers the ownership of a file from one user to another.
DUNPRG	A1-23	Creates a file and moves the contents of a disc site into it.
GET AV FSYS	A1-22	Gives the numbers of all the disc-packs currently on-line.
GET USNAMES	A1-22	Gives the names of the users whose files and file indexes are on a disc-pack.
OPER	A1-41	(system routine) Causes a text string to be displayed on an Operator's console.
PRIME CONTINGENCY	A1-31	Nominates a routine to be executed on the occurrence of a contingency.
READ ID	A1-33	Gives the process's environment at the most recent contingency.

A1.3 Introduction to the file system

This section is intended to be an initial guide to the file system, but does not comprise a detailed description. The file system closely follows the philosophy of that for System 4 EMAS (see Ref. 7). The main principles and features are as follows:

a) Each disc-pack is treated as a unit; all the files (and the file index) for a given user reside on one disc-pack.

b) The file system is implemented "in process" and comprises provision of the following primitives, to be called by outer code (higher ACR, less privileged), in particular a subsystem:

   CREATE file

   CHANGE file SIZE

   DESTROY file

   CONNECT file

   DISCONNECT file

   Interfaces to lower ACR code are limited to the following requests:

   * MOVE or CLEAR section of disc space

   * CLAIM or FREE semaphore number conventionally associated with a file index or "bitmap".

   * For CONNECT and DISCONNECT, writing into or deleting from the "Master Page Tables" the relationships between the segment numbers of the virtual memory and groups of sections of disc space. The "Master Page Tables" reside in the Local Controller stack for the process, and the Local Controller organises the virtual memory hierarchy and the satisfying of page faults from the information therein.

c) Just as the user accesses a file by requesting that it be CONNECTed into his virtual memory and then referencing the appropriate virtual addresses obtained, so the Director code, which implements the file system, creates and accesses the indexes and the "bitmaps" by CONNECTing them also into the virtual memory at a lower ACR level (more privileged) than that of the subsystem and the user.

A1.3.1 File space allocation and index maintenance

Director organises the file system in terms of epage units (see the Glossary in Ref. 2). A 100 Mbyte disc pack contains 24000 epages (approximately X'5E00'), and currently Director uses from X'40' or X'800' to X'5000' for the file system. (On a disc which may be used as a System disc, epages below X'800' are used for the IPL supervisor (Chopsupe), 3 versions of a main supervisor, 4 versions of Director, and so on.) The first 128 pages of the file system space are used for the "bitmap" for the disc pack, and for the file indexes (currently 1 or 2 epages each) for the users whose files reside on the disc-pack.

The "bitmap" is an area containing one bit representing each epage on the pack, numbered from 0 to X'5E00' approximately. A zero bit means that the epage is free, a one bit that it is allocated. The bitmap is located at the start of the file system space, the indexes follow, and the users' file pages start beyond the indexes. The actual start page numbers are all constants in Director's code.

Director allocates the users' files in maximum-sized units of one "section", currently 16 epages or 64 Kbytes. A file consisting of 1 segment has up to 4 sections (i.e. 64 epages or 256 Kbytes), and larger files have more sections as necessary. The file index contains the string name of the file, and a file descriptor in the index contains a chain of list cells containing the starting epage number of each section of the file. When a CREATE request is made, Director searches the bitmap for a group or groups of the required numbers of free pages and sets the bits accordingly. For a CONNECT request, Director extracts the starting epage numbers of the sections of the file from the index and inserts them into the "claimed block" table and puts pointers from the "secondary segment" table to the claimed block table entries. The addresses of the starts of these tables are passed to Director at process start-up.

Users' file indexes are numbered according to the epage number at which they start; currently each index comprises one or two epages. A file system request from a user or subsystem specifies the file <u>owner</u> (6 characters) and the file <u>name</u> (up to 11 characters). In order to reference the file index for the owner, Director first connects it into the virtual memory. To do this, Director first searches the "name-number table", which relates the owner names of the indexes on the pack to the index starting page numbers. The name-number table currently follows the bitmap on the disc. When the entry for the file owner has been found, the corresponding number indicates the disc section containing the file index; Director then connects it into the virtual memory.

N.B. Director connects bitmaps, name-number tables and indexes "on demand" into the user's virtual memory in segments (not accessible to the user) numbered between 16 and 31. For efficiency, the bitmap and name-number table for the pack on which the process owner's file index and files reside, and also the section containing the file index itself, are left connected for the duration of the process.

A1.4 File system procedures

<u>externalintegerfn</u> DCREATE (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>integer</u> FSYS, NKB, TYPE)

A file of name FILE is created, for user USER on disc-pack FSYS, of E Epages, where E is the smallest number of Epages containing NKB Kbytes.

The maximum size of file allowed is 16 Mbytes. Subsystems requiring larger files should arrange that they be made up of subfiles comprising files created by this procedure.

Note 1 applies.
Note 2 applies.

Bits in TYPE may be set:

2**0	For a temporary file (destroyed when the creating process stops if the file was connected, or at System start-up).
2**1	For a very temporary file (destroyed when the file is disconnected).
2**2	For a file which is to be zeroed when created.
2**3	To set "CHERISHed" status for the file.

Temporary files may be created only in the process owner's file index, and may be connected only into a virtual memory of the process owner. In particular, temporary files may not be shared.

Temporary files are made into ordinary files (that is, the "temporary" attribute is removed) on being RENAMEd, OFFERed, TRANSFERred or PERMITted, and also explicitly by an appropriate call on procedure DFSTATUS.

Possible error results: 11, 15, 16, 17, 18, 26, 28, 37, 41, 47

****

<u>externalintegerfn</u> DDESTROY (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>string</u>(8) DATE, <u>integer</u> FSYS, TYPE)

The file, of name FILE belonging to user USER on disc-pack FSYS, is destroyed. TYPE should be set to 1 to destroy a file from archive storage; otherwise it should be set to zero. When TYPE=1, DATE should be set to the archive date. DATE is ignored if TYPE=0.

Note 1 applies.
Note 2 applies.
Note 3 applies.

The procedure fails if the file owner has made access permission zero for 'self'.

Possible error results: 5, 8, 11, 18, 20, 21, 22, 25, 32, 37, 47, 59, 75

****

externalintegerfn DCONNECT (string(6) USER, string(11) FILE, integer FSYS, MODE, APF, integername SEG, GAP)

Provided that the file is suitably permitted to the process owner calling the procedure, the file of name FILE belonging to user USER on disc-pack FSYS is connected into the caller's virtual memory.

Note 2 applies.

The bits in the parameter MODE have the following meanings (when set):

2**0	Read access required
1	Write access required
2	Execute access required
3	Write access by other processes to be allowed
4	New copy of file to be written
5	Communications mode
6	Not to be allocated space on the drum
7	Segment to be used as a process stack

The purpose of bit 2**3 is to allow (read and) write access by more than one process to be achieved only when each user specifically allows the situation (by setting the bit in his request).

Bits 2**1 or 2**3 may not be set in the request if bit 2**2 (execute access) is also set.

SEG either specifies the segment number at which the file is to be connected (in the range 34 to 127), or is zero, indicating that the choice of segment number is to be left to Director. If the result of the function is 0 or 34 (file already connected), SEG is set to the chosen segment number.

GAP specifies the number of segments which are to be reserved for the file, even though the current size of the file may be less than that number of segments. Attempts to specify a value of SEG which conflicts with this GAP, in subsequent connect requests before this file is disconnected, will be rejected. If GAP is set to zero then no segments of virtual memory, other than those required by the current file size, are reserved for the file. If the result of the function is 0 or 34 (file already connected), GAP is set to the number of segments reserved for the file.

APF may be used to specify the access permission field in the segment(s) being connected. The bottom 9 bits are significant:

1	4	4
EXE-CUTE	WRITE ACR	READ ACR

The read and write ACR values supplied must be greater than or equal to the ACR at which the calling program (subsystem) is running. If the APF parameter is set to zero, a value of X'1nn' is used, where n is the ACR at which the caller is executing.

Possible error results: 5, 6, 26, 28, 30, 32, 34, 35, 36, 37, 47

****

<u>externalintegerfn</u> DDISCONNECT (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>integer</u> FSYS, DESTROY)

The file of name FILE belonging to user USER on disc-pack FSYS is disconnected from the caller's virtual memory. Parameter DESTROY should be set either to 0 or 1. If set to 1 the file will be destroyed, provided that it belongs to the process owner (not necessary if the process is privileged) and the "use-count" for the file is zero after disconnection. Otherwise the parameter is ignored.

Note 2 applies.

Possible error results: 30, 32, 37, 38, 39, 47

****

<u>externalintegerfn</u> DCHSIZE (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>integer</u> FSYS, NEWSIZE)

The physical size of file FILE belonging to user USER on disc-pack FSYS is altered (if necessary) so that its new size (in Kbytes) is NEWSIZE. The size may not be reduced to zero. The file may be connected in the caller's virtual memory (only).

Note 1 applies.
Note 2 applies.

Possible error results: 11, 30, 32, 37, 41

****

```
externalintegerfn DRENAME (string(6) USER, string(11) OLDNAME, c
                             NEWNAME, integer FSYS)
```

The file of name OLDNAME belonging to user USER on disc-pack FSYS is renamed NEWNAME.

Note 1 applies.
Note 2 applies.

A file may not be renamed while it is connected in any virtual memory.

Possible error results: 5, 11, 16, 17, 18, 32, 40

****

```
externalintegerfn DOFFER (string(6) USER, OFFERTO, string(11) FILE, c
                          integer FSYS)
```

This procedure causes file FILE belonging to user USER on disc-pack FSYS to be marked as being "on offer" to user OFFERTO. The file may not be connected in any virtual memory either at the time of the call of this procedure or subsequently while the file is on offer. The procedure DACCEPT is used by user OFFERTO to accept the file from user USER. A file may be on offer to at most one user. An offer may be withdrawn by calling this procedure with OFFERTO set as a null string.

Note 1 applies.
Note 2 applies.

Possible error results: 11, 17, 20, 30, 32

****

```
externalintegerfn DACCEPT (string(6) USER, string(11) FILE, NEWNAME, c
                             integer FSYS)
```

This procedure causes the transfer to the caller of ownership of file FILE belonging to user USER on disc-pack FSYS. The file must previously have been "offered to" the caller of this procedure, using procedure DOFFER. The file is named NEWNAME under its new ownership, but NEWNAME and FILE may be identical names.

Note 1 applies.
Note 2 applies.

Possible error results: 5, 8, 9, 15, 16, 17, 18, 32, 37

****

```
externalintegerfn DPERMISSION (string(6) OWNER, USER, string(8) DATE, c
                               string(11) FILE, integer FSYS, TYPE, ADRPRM)
```

This function allows the owner of file FILE on disc-pack FSYS to set access permissions, or specific preventions, for file connection to individual users, groups of users or to all users. It also allows a user to determine the modes (if any) in which he may access the file.

Note 2 applies.
Note 3 applies.

TYPE determines the service required of the procedure:

TYPE	Action
0	set OWNP (not allowed for files on archive storage)
1	set EEP
2	put USER into the file list (see "Use of file access permissions", below)
3	remove USER from file list
4	return the file list
5	destroy the file list
6	put USER into the index list (see "Use of file access permissions", below)
7	remove USER from the index list
8	return the index list
9	destroy the index list
10	give modes of access available to USER for FILE

TYPEs 0 to 9 are available only to the file owner and to privileged processes. For TYPE 10, ADRPRM (see below) should be the address of an integer into which the access permission of USER to the file is returned. If USER has no access to the file, error result 32 will be returned from the function, as though the file did not exist. If the file is on archive storage, TYPE should be set to 16 plus the above values to obtain the equivalent effects.

ADRPRM is either the permission being attached to the file, bit values interpreted as follows:

all bits zero	prevent access	
2**0	allow READ access	
2**1	allow WRITE access	not allowed for files
2**2	allow EXECUTE access	on archive storage

<u>or</u>, except for type 10, it is the address of an area into which access permission information is to be written, in the format

```
  (integer BYTES RETURNED, OWNP, EEP, SPARE, recordarray c
       INDIV PRMS(0:15) (string(6) USER, byteinteger UPRM))
```

where:

BYTES RETURNED	indicates the amount of data returned.
OWNP	is the file owner's own permission to the file, or the requesting user's "net" permission if the caller of the procedure is not the file owner (see "Use of file access permissions", below).
EEP	is the general (all users) access permission to the file ("everyone else's permission").
UPRM	The UPRM values in the sub-records are the permissions for corresponding users or groups of users denoted by USER. Up to 16 such permissions may be attached to a file.

Use of file access permissions

The general scheme for permissions is as follows. With each file there are associated:

OWNP	the permission of the owner of the file to access it
EEP	everyone else's permission to access it (other than users whose names are explicitly or implicitly attached to the file)
INDIV PRMS	a list of up to 16 items describing permissions for individual users, e.g. ERCC00, or groups of users, e.g. ERCC?? (specifying all usernames of which the first four characters are "ERCC")

In addition, a user may attach a similar list of up to 16 items to his file index <u>as a whole</u>, and the permissions in this list apply to any file described in the index along with those attached to that particular file.

In determining the mode or modes in which a particular user may access a file, the following rules apply:

1. If the user is the file owner then OWNP applies.

2. Otherwise, if the user's name appears explicitly in the list for the file, the corresponding permission applies.

3. Otherwise, if the user's name appears explicitly in the list for the index, the corresponding permission applies.

4. Otherwise, if the user's name is a member of a group of users represented by a list item for the file, the corresponding permission applies.

5. Otherwise, if the user's name is a member of a group of users represented by a list item for the index, the corresponding permission applies.

6. Otherwise EEP applies.

In the event of a user's name appearing more than once (implicitly) within groups specified in a single list, the actual list item to be selected to give the permission should be regarded as indeterminate.

Possible error results: 8, 11, 17, 32, 37, 45, 46, 59, 75

****

externalintegerfn DFINFO (_string_(6) USER, _string_(11) FILE, _c_
integer FSYS, ADR)

This procedure returns detailed information about the attributes of file FILE belonging to user USER on disc-pack FSYS, in a record written to address ADR.

Note 2 applies.

A non-privileged caller of the procedure having no permitted access to the file will receive an error result of 32, as though the file did not exist.

The format of the record returned is of successive integers and a _string_(6), as detailed below:

NKB	the number of Kbytes (physical file size)
RUP	the caller's permitted access modes
EEP	the general access permission
APF	1-4-4 bits, right-justified, giving respectively the Execute, Write and Read fields of APF, if the file is connected in this virtual memory
USE	the current number of users of the file
ARCH	the value of the archive byte for the file (see procedure DFSTATUS, below)
FSYS	disc-pack number on which the file resides
CONSEG	the segment number at which the file is connected in the caller's virtual memory; zero if not connected
CCT	the number of times the file has been connected since the connect count was last zeroed (see procedure DFSTATUS)

CODES	information for privileged processes (see the diagram in Subsection A1.4.2)
CODES2	information for internal use (see the diagram in Subsection A1.4.3)
SSBYTE	information for the subsystem's exclusive use
string(6) OFFER	the username to which the file has been offered (see procedure DOFFER); otherwise null

Possible error results: 18, 32, 37, 45

****

```
externalintegerfn DFSTATUS (string(6) USER, string(11) FILE, c
                            integer FSYS, ACT, VALUE)
```

This procedure is supplied to enable the attributes of file FILE belonging to user USER on disc-pack FSYS to be modified, as follows.

Parameter VALUE is for use by the archive/backup program (ACT=13), and by the subsystem (ACT=18); otherwise it should be set to zero.

The layout of a file's archive byte is shown in the diagram in Subsection A1.4.1.

Note 1 applies.
Note 2 applies.

ACT		ACTION
0	HAZARD	Remove CHERISHed attribute.
1	CHERISH	Make subject to automatic System back-up procedures.
2	UNARCHIVE	Remove the "to-be-archived" attribute.
3	ARCHIVE	Mark the file for removal from on-line to archive storage.
4	NOT TEMP	Remove the "temporary" attribute.
5	TEMPFI	Mark the file as "temporary"; that is, to be destroyed when the process belonging to the file owner stops (if the file is connected at that time), or at system start-up.
6	VTEMPFI	Mark the file as "very temporary"; that is, to be destroyed when it is disconnected from the owner's virtual memory.

7	NOT PRIVATE	May now be written to magnetic tape either for back-up or archive. May be called only by privileged programs.
8	PRIVATE	Not to be written to magnetic tape either for back-up or archive. May be called only by privileged programs.
9	SET CCT	Set the connect count for the file to the bottom 8 bits of VALUE.
10	ARCH	Operation 0 (PRIVILEGED). Shift ARCH byte usage bits (2**2 to 2**6 inclusive) left one place. If A is the resulting value of the ARCH byte, set bit 2**7 if (A>>2)&B'11111' = VALUE.
11	ARCH	Operation 1 (PRIVILEGED). Set currently-being-backed-up bit (bit 2**1 in ARCH byte), unless the file is currently connected in write mode, when error result 52 is given.
12	ARCH	Operation 2 (PRIVILEGED). Clear currently-being-backed-up bit (2**1) and has-been-connected-in-write-mode bit (2**0).
13	ARCH	Operation 3 (PRIVILEGED). Set archive byte to be bottom 8 bits of VALUE and clear the UNAVAilable bit in CODES.
14	ARCH	Operation 4 (PRIVILEGED). Clear the UNAVAilable and privacy VIOLATed bits in CODES. Used by the back-up and archive programs when the file has been read in from magnetic tape.
15	CLR USE	Clear file use-count and WRITE-CONNECTED status (PRIVILEGED).
16	CLR NOARCH	Clear archive-inhibit bit in CODES. } PRIVILEGED - for System Library use
17	SET NOARCH	Set archive-inhibit bit in CODES. } PRIVILEGED - for System Library use
18	SSBYTE	Set SSBYTE to be the bottom 8 bits of VALUE (byte for a subsystem's exclusive use).
19	ARCH	Operation 5 (PRIVILEGED). Set the WRCONN bit in CODES2. Used to prevent any user connecting the file in write mode during back-up or archive.
20	ARCH	Operation 6 (PRIVILEGED). Clear the WRCONN bit in CODES2. Used when back-up is complete.

Possible error results: 8, 11, 18, 20, 37, 47, 52

****

externalintegerfn DSFI (_string_(6) USER, _integer_ FSYS, TYPE, SET, ADR)

(Etymology: SFI - System File Information)

This procedure is used to set or read information in the file index of USER on disc-pack FSYS. TYPE specifies which data item is to be referenced (see list below). SET must be 1 to write the data item into the index, or 0 to read the item from the index. ADR is the address of an area, which must be available in write or read mode, to or from which the data item is to be transferred.

Note 1 applies.
Note 2 applies.

TYPE	Data item	Data type & size
0	BASEFILE name (the file to be connected and entered at process start-up)	_string_(31)
1	DELIVERY information (to identify slow-device output requested by the index owner)	_string_(31)
2	CONTROLFILE name (a file for use by the subsystem for retaining control information)	_string_(31)
3	ADDRTELE address and telephone number of user	_string_(63)
4	INDEX USE (may not be reset) Gives (in successive integers from ADR): a) number of files b) number of file descriptors currently in use c) number of free bytes currently available for file descriptors (allow say 32 per new file descriptor) d) index size (Kbytes) e) 4 (maxcells, freecells) pairs, each comprising 2 integers; unused pairs are zero	12x_integer_
5	Foreground and background passwords (reading is a privileged operation); a zero value means "do not change"	2x_integer_
6	Date last logged-in: (Y-70)<<9 ! (M<<5) ! D (may not be reset)	_integer_
7	ACR level at which the process owning this index is to run (may be set only by privileged processes)	_integer_
8	Director Version (may be set only by privileged processes)	2x_integer_
9	ARCHIVE INDEX USE (may not be reset) Gives (in successive integers from ADR): a) number of files b) number of file descriptors currently in use c) number of free bytes currently available for file descriptors (allow say 32 per new file descriptor)	

	d) index size (Kbytes) e) 4 (maxcells, freecells) pairs, each comprising 2 integers; unused pairs are zero	12xinteger
10	Stack size (Kbytes)	integer
11	Limit for total size of all files in disc storage (Kbytes) (may be set only by privileged processes	integer
12	Maximum file size (Kbytes) (may be set only by privileged processes)	integer
13	Current numbers of interactive and batch processes, respectively, for the user (may not be reset)	2xinteger
14	Process concurrency limits (may be set only by privileged processes). The three words denote respectively the maximum number of interactive, batch and total processes which may be concurrently running for the user. (Setting the fields to -1 implies using the default values, currently 1, 1 and 1.)	3xinteger
15	Privacy of presence and message control (see Section A1.8) Bits in this word have the following meanings when set: 2**0 process presence may be detected by other users 2**1 allow incoming text messages to be received 2**2 signal arrival of incoming messages via the contingency mechanism	integer
16	Set Director monitor level (may be set only by privileged processes)	2xinteger
17	Set SIGNAL monitor level (may be set only by privileged processes)	integer
18	Initials and surnames of user (may be set only by privileged processes)	string(31)
19	Director monitor file	string(31)

METERING INFORMATION (cumulative over process sessions except where shown)

20	Thousands of instructions executed, interactive and batch modes (may be reset only by privileged processes)	2xinteger
21	Thousands of instructions executed (current session only)	integer
22	Thousands of instructions executed in Director procedures (current process session only) (may not be reset)	integer

23	Page-turns, interactive and batch modes (may be reset only by privileged processes)	2xinteger
24	Page-turns (current process session only) (may be reset only by privileged processes)	integer
25	Thousands of bytes output to slow-devices (local or remote) (may be reset only by privileged processes)	integer
26	Thousands of bytes input from slow-devices (local or remote) (may be reset only by privileged processes)	integer
27	Milliseconds of OCP time used, interactive and batch modes (may be reset only by privileged processes)	2xinteger
28	Milliseconds of OCP time used (current session only)	integer
29	Seconds of interactive terminal connect time (may be reset only by privileged processes)	integer
30	No. of disc files, total disc Kbytes, no. of cherished files, total cherished Kbytes, no. of temporary files, total temporary Kbytes (cannot be reset)	6xinteger
31	No. of archive files, total archive Kbytes	2xinteger
32-34	Spare	
35	Test BASEFILE name	string(31)
36	Batch BASEFILE name	string(31)

Possible error results: 8, 37, 45

****

externalintegerfn DFSYS (string(6) USER, integername FSYS)

This ~~privileged~~ procedure may be used to determine on which dick-pack user USER resides. If FSYS is set to -1 before the procedure is called, it is set with the first disc-pack number on which USER is found. If FSYS is set non-negative, only that disc-pack number is searched. If USER is not found, FSYS is unchanged and error result 37 is returned.

Possibkle error results: 8, 23, 37

****

```
externalintegerfn DNEWGEN (string(6) USER, string(11) FILE, c
                          NEWGEN OF FILE, integer FSYS)
```

This procedure provides a means of introducing an updated version (i.e. a new generation) of file FILE even though it may be connected in other users' virtual memories.

If FILE is not connected in any virtual memory, a call on DNEWGEN is equivalent to destroying FILE and then renaming NEWGEN OF FILE to FILE, except that the new version of FILE retains the former FILE's access permissions.

If FILE _is_ connected in some virtual memory, then the filename NEWGEN OF FILE "disappears", and any subsequent connection of FILE into a virtual memory yields the contents of the new generation formerly held in NEWGEN OF FILE.

When the number of users of a former copy of FILE becomes zero (i.e. when it is not connected in any virtual memory), that copy is destroyed.

Note 1 applies.
Note 2 applies.

Possible error results: 5, 6, 11, 15, 18, 32, 37, 40

****

```
externalintegerfn DFILENAMES (string(6) USER, recordarrayname INF, c
                             integername FILENUM, MAXREC, NFILES, c
                             integer FSYS, TYPE)
```

This procedure delivers, in the record array INF (which should be declared (0:n)), a sequence of records describing the on-line files (for TYPE=0) or archived files (for TYPE=1) belonging to USER.

Note 1 applies.
Note 2 applies.
Note 3 applies.

MAXREC is set by the caller to specify the maximum number of records he is prepared to accept in the array INF, and is set by Director to be the number of records actually returned.

NFILES is set by Director to be the number of files actually held on on-line storage _or_ on archive storage, depending on the value of TYPE.

Parameter FILENUM is used only for TYPE=1. Filenames are stored in chronological order (by archive date). FILENUM is set by the caller to specify the "file-number" from which descriptions are to be returned; zero represents the most recently archived file. (The intention here is to allow the caller to receive subsets of descriptions of a possibly very large number of files.)

The format of the records delivered in the array INF is as follows:

(string(11) NAME, integer SP12, KBYTES, byteinteger ARCH, CODES, c
 CCT, OWNP, EEP, USE, CODES2, SSBYTE, FLAGS, SP29, SP30, SP31)
(32 bytes)

for on-line files, and

(string(11) NAME, integer KBYTES, string(8) DATE, string(6) TAPE, c
 integer CHAPTER, FLAGS)
(40 bytes)

for archived files. TAPE and CHAPTER are returned null to unprivileged callers.

Possible error results: 11, 37, 45, 59, 61

****

externalintegerfn DTRANSFER (string(6) USER1, USER2, string(11) c
 FILE, NEWNAME, integer FSYS1, FSYS2, TYPE)

This procedure is available only to privileged processes, and transfers FILE belonging to user USER1 on disc-pack FSYS1 to the ownership of user USER2 on disc-pack FSYS2 under name NEWNAME. TYPE should be set to 1.

Possible error results: 3, 5, 8, 15, 16, 17, 18, 32, 37

****

externalintegerfn DGETDA (string(6) USER, string(11) FILE, c
 integer FSYS, ADR)

This procedure, available only to privileged processes, provides the disc addresses of the sections of file FILE belonging to USER on disc-pack FSYS. Data is written from address ADR in the form

(integer SECTSI, NSECTS, LASTSECT, SPARE, integerarray DA(0:255))

where	SECTSI	is the size (in epages) of the sections (except possibly the final section)
	NSECTS	is the number of sections, and hence the number of entries returned in array DA
	LASTSECT	is the size (in epages) of the final section

In each entry in the DA array, the top byte contains the FSYS number.

Possible error results: 5, 30, 32, 37, 45

****

externalintegerfn GET USNAMES (recordarrayname NN, integername N, c
integer FSYS)

This procedure, available only to privileged processes, supplies the list of usernames accreditted on disc-pack FSYS. A series of records of format

(string(6) NAME, byteinteger KB, integer INDNO)

is returned in the array NN, which should be declared (0:511). N is set to the number of names supplied, a likely maximum being 512.

Possible error results: 23, 45

****

externalroutine GET AV FSYS (integername N, integerarrayname A)

This procedure, available only to privileged processes, supplies the logical disc-pack numbers of disc-packs currently on-line. Array A, which should be declared (0:63), is filled from A(0), A(1), ... with as many numbers as there are on-line EMAS disc-packs, and N is set to the number of entries returned.

Note that, when a disc-pack is mounted, it is not considered "on-line" until a file system consistency check has been performed on it.

****

externalintegerfn DPRG (string(6) USER, string(11) FILE, c
integer FSYS, string(6) LABEL, integer SITE)

This procedure, available only to privileged processes, moves the contents of file FILE belonging to user USER on disc-pack FSYS to site SITE on the EMAS 2900 disc-pack labelled LABEL.

SITE is an epage number which must be X'40'-aligned. The physical size of the file must not exceed 256 Kbytes (512 Kbytes for sites X'300' and X'380').

Note 2 applies.

Possible error results: 2, 5, 27, 30, 32, 37

****

<u>externalintegerfn</u> DUNPRG (<u>string</u> (6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>integer</u> FSYS, <u>string</u>(6) LABEL, <u>integer</u> SITE)

This procedure is available only to privileged processes. It creates a 256 Kbyte file FILE belonging to user USER on disc-pack FSYS and copies into it 256 Kbytes from site SITE on the EMAS 2900 disc-pack labelled LABEL.

Note 2 applies.

Possible error results: 2, 5, 16, 27, 30, 32, 37

****

<u>externalintegerfn</u> ACREATE (<u>string</u>(6) USER, TAPE, <u>string</u>(8) DATE, <u>c</u>
<u>string</u>(11) FILE, <u>integer</u> FSYS, NKB, CHAPTER)

This procedure is provided for use by the archive program. A new archive index entry is created for USER, giving TAPE, CHAPTER and no-of-Kbytes attributes to be associated with FILE. (Access permission attributes are given to FILE by separate calls of DPERMISSION.)

DATE should normally be left null, when the current date will be used.

Possible error results: 8, 15, 16, 37, 59, 75

****

<u>externalintegerfn</u> DNEW ARCH INDEX (<u>string</u>(6) USER, <u>integer</u> FSYS, KBYTES)

This privileged procedure is for use by the System Manager, and creates a new archive index of SIZE Kbytes for user USER on disc FSYS. The minimum size allowed is 4 Kbytes, allowing about 80 archive files to be described.

Possible error results: 8, 12, 15, 16, 17, 23, 37

****

<u>externalintegerfn</u> DRESTORE (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>string</u>(8) DATE, <u>integer</u> FSYS, TYPE)

This procedure passes a restore request (if FILE exists on archive storage and is permitted to the caller) to VOLUMS. DATE may be left null, when the most recently archived copy of FILE will be restored. The file is restored into the file owner's on-line index. TYPE is currently ignored and should be set to zero.

Possible error results: 8, 15, 16, 32, 37, 59, 75

****

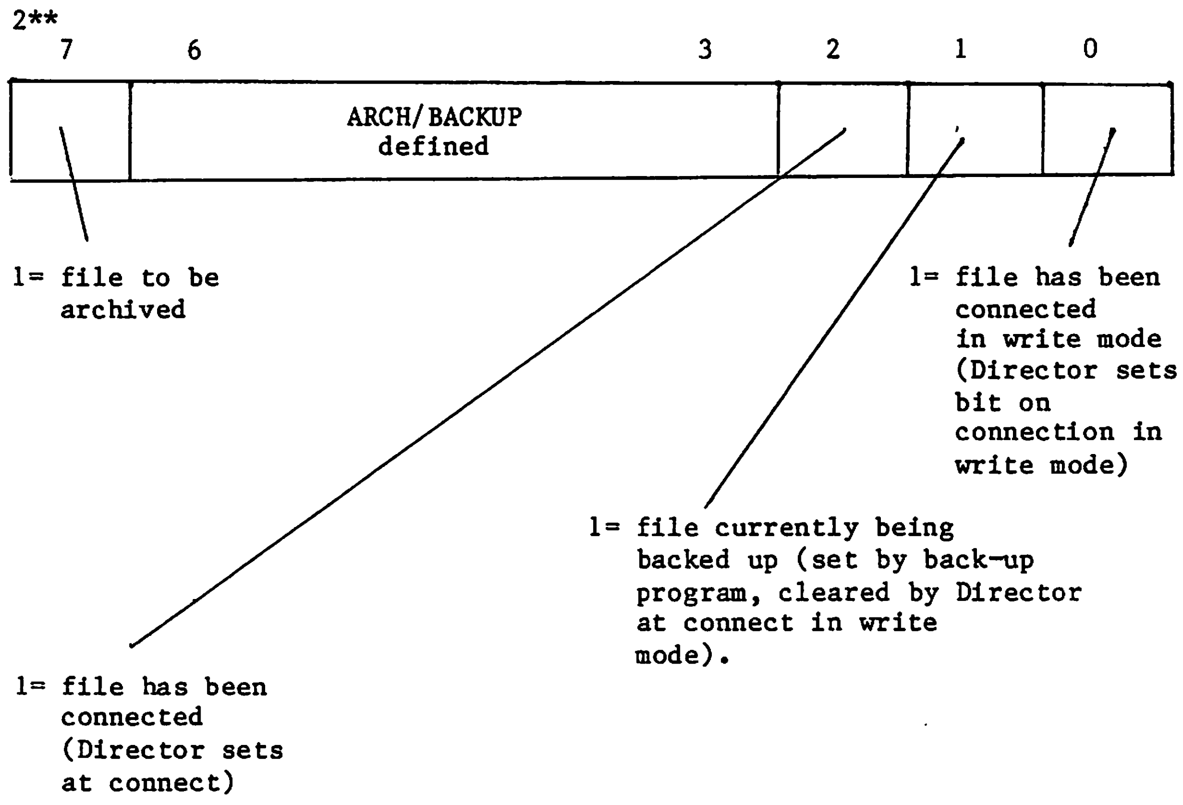
<u>externalintegerfn</u> DMODARCH (<u>string</u>(6) USER, <u>string</u>(11) FILE, <u>c</u>
<u>string</u>(8) DATE, <u>recordname</u> ENT, <u>integer</u> FSYS)

This privileged procedure is provided for the System Manager to make amendments to archive index entries. USER, FILE, DATE and FSYS determine the entry to be modified. Record ENT has the same format as that supplied by DFILENAMES (TYPE=1). Fields (other than NAME) which differ from those of the specified index entry will be used to update the entry.

Possible error results: 8, 15, 16, 37, 59, 75

****

A1.4.1 <u>Contents of ARCHIVE byte</u>


2**
7
6
3
2
1
0
ARCH/BACKUP defined
1= file to be archived
1= file has been connected in write mode (Director sets bit on connection in write mode)
1= file currently being backed up (set by back-up program, cleared by Director at connect in write mode).
1= file has been connected (Director sets at connect)

A1.4.2 Contents of CODES byte

(All bits are of interest to back-up/archive program)

0	1	2	3	4	5	6	7
ARCH INHIB	VIOLAT	PRIVAT	CHERS	VTEMP	TEMPF	OFFER	UNAVA

A1.4.3 Contents of CODES2 byte

0	1	2	3	4	5	6	7
DEAD	STACK	DISC ONLY	COMMS MODE	WS ALLOW	OLDGEN	NEWGEN	WRCONN

```
externalintegerfn DENABLE TERMINAL STREAM (string(11) FILE, integer c
                     STREAM, MODE, LEVEL, OFFSET, LEN, integername SEG)
```

When a user logs in to EMAS 2900 from an interactive terminal, a process is created and two interactive streams, one for input and one for output, connect the process to the terminal. (Note that this is a usage of the word "connect" distinct from that pertaining to files and virtual memories in EMAS 2900.) The process sends output to its terminal by writing the data into a file which has been connected into its virtual memory in a special "communications output mode". Similarly, input from the terminal is placed by the System in another file which has been connected in "communications input mode". In each case, the file (or part of it) is used by the process and the communications system as a circular buffer.

Procedure DENABLE TERMINAL STREAM is used to connect file FILE into the virtual memory and to specify the circular buffer within the file: OFFSET gives the offset (in bytes) of the start of the buffer from the start of the file, and LENGTH gives the length (in bytes) of the buffer. STREAM should be set to 0 or 1, indicating an input or output file respectively. MODE and LEVEL are currently not used and should be set to zero. SEG may be set to determine the segment number at which the file is to be connected, or may be set to 0 if Director is to choose a segment (cf. Director procedure DCONNECT), in which case it is set with the segment number on return from the procedure.

After this procedure has been successfully executed, the specified circular buffer is available for interactive input or output. After a stream has been disabled or aborted (see procedure DDISABLE TERMINAL STREAM), this procedure may be called again to re-specify a circular buffer (FILE will be already connected), to allow input or output to be resumed.

Each EMAS 2900 process may have at most one input and one output stream connected to the communications system.

Possible error results: 5, 6, 26, 28, 30, 32, 35, 36, 37, 47

****

```
externalintegerfn DREQUEST TERMINAL OPERATION (integer STREAM, c
                                         OUTPUT POSN, TRIGGER POSN)
```

(Note that the parameters OUTPUT POSN and TRIGGER POSN refer throughout to byte offsets within the relevant circular buffer.)

Input (STREAM=0)

When STREAM is set to 0, the procedure is used to request input and to specify how much of the existing data has been read: i.e. up to, but not including, the byte specified by TRIGGER POSN. TRIGGER POSN thus gives an upper limit on the extent of new input, the buffer being circular.

The unit of input information is the message. The delimiter of a message is determined by options set in the Terminal Control Processor (TCP), but is commonly line-feed (LF, ISO decimal 10).

In the first input call of DREQUEST TERMINAL OPERATION, TRIGGER POSN must be specified as 0. The communications system places input messages in the buffer from the start of the buffer, and continues to do so, autonomously with respect to the subsystem, so long as there are complete input messages to pass on and there is free space in the buffer.

After each complete message has been transferred to the buffer, the communications system updates a pointer to the byte beyond the last byte written. This pointer can be read, but not changed, by the subsystem; it may be referenced by declaring

```
          constintegername INPOINTER = X'00140050'
```

Then the use of INPOINTER as an identifier in an IMP statement causes a reference to the required word; e.g.

```
          if NEXT = INPOINTER start
               .
               .
```

From the subsystem's point of view, on return from an input call of DREQUEST TERMINAL OPERATION, unprocessed input messages are available in the buffer from the TRIGGER POSN specified in the call, up to (but not including) the position pointed at by the read-only pointer. The latter may change while the subsystem is processing the input, since further messages may be transferrred at any time to the input buffer by the communications system.

In general, the subsystem will only call the procedure to request input when it has processed all the existing information in the input buffer, as indicated by the read-only pointer. As part of such an input request, the subsystem may specify a character string, to be output at the interactive terminal if no input has been typed by the user by the time that the request reaches the TCP. This string is called the "prompt" string, and is restricted to 15 characters maximum by currently implemented TCP software. The subsystem places the required prompt in the output circular buffer, beyond the last byte of data output by the last output call of DREQUEST TERMINAL OPERATION (see below). In an input call of the procedure, the subsystem specifies, via OUTPUT POSN, the position of the byte following the last byte of the prompt string in the output buffer, and this is sufficient to inform the communications system of the existence and position of the prompt string. (Subsequent output by the subsystem will overwrite this prompt string, since the communications system's output buffer pointer is not updated as part of the prompt operation.)

If, following an input call of the procedure, there are no input messages, the prompt is printed on the user's terminal and the process "goes to sleep" until the next complete input message is placed in the buffer; when this happens the read-only pointer is updated and the process reawakened.

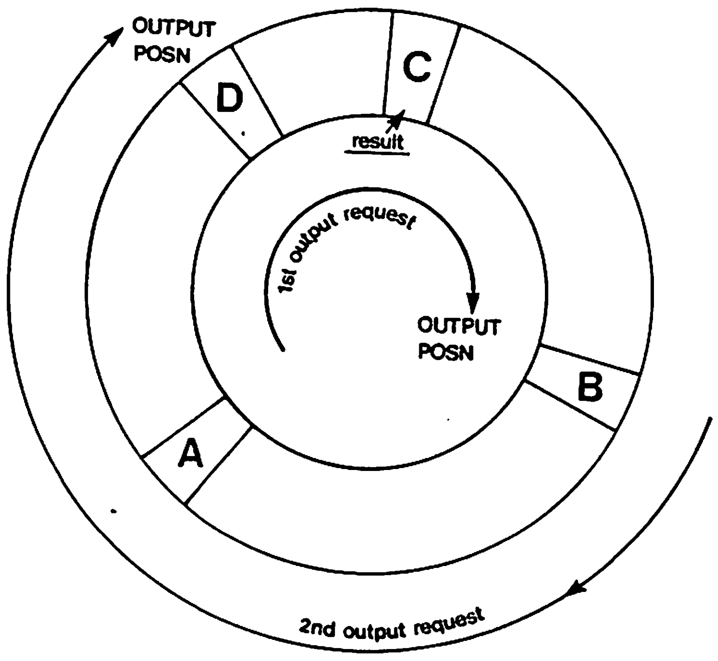
If an input message were received by the communications system and transferred to the buffer between the time that the subsystem last examined the read-only pointer and the time that the procedure was activated, then an immediate return would be made from the procedure to the subsystem.

Output (STREAM=1)

To send output to the interactive terminal, the subsystem must call DREQUEST TERMINAL OPERATION with STREAM set to 1. On the first output request, bytes are dispatched from the start of the circular output buffer. In all output calls of the procedure (including the first), OUTPUT POSN is set by the subsystem to indicate the byte after the latest byte written in the buffer to date.

Control will not be handed back from DREQUEST TERMINAL OPERATION (i.e. the process will be suspended) until data have been extracted from the buffer up to at least the byte before the byte indicated by the parameter TRIGGER POSN. When control is handed back, the <u>result</u> of the procedure gives the position of the last byte actually transferred out of the buffer by the communications system to date. This <u>result</u> can thus be used by the subsystem to ensure that it does not overwrite data in the circular buffer before it has been dispatched. Following the return, the communications system will continue to transfer data out of the buffer, autonomously with respect to the subsystem, until the specified OUTPUT POSN has been reached.

As an example of the use of the facilities provided by this mechanism, consider the following:



The first output request specifies the last byte written as being the byte before position B, the communications system having previously dispatched data up to position A. The result of the first output request is position C. The subsystem places subsequent output data in the circular output buffer starting at position B; this data can extend as far as position C - the subsystem cannot use more as it does not know when the rest of the first output request is carried out by the communications system. In the diagram, the OUTPUT POSN in the second output request is position D, which is before position C in the buffer.

The TRIGGER POSN value in the first output request must clearly have indicated a point no further down the buffer than the byte after position C. In the second output request, TRIGGER POSN could specify any byte after position C up to and including position D.

TRIGGER POSN may also be set to -1 by the subsystem, in which case an immediate return from the procedure will always be given. As always, the subsystem simply needs to check that it does not write data into the buffer beyond the position indicated by the result of the procedure.

Other interesting possible values of TRIGGER POSN are: 1) equal to OUTPUT POSN, when control will only be returned when all output data have been taken from the buffer; 2) 1 more than the previous value of OUTPUT POSN (i.e. the first byte of the current request). In the latter case processing is unlikely to be held up, since it would imply some serious blockage of the communications system.

Possible error result: -1 (illegal parameter or "POSNs")

Note that a positive result from an output request is not an error result.

****

externalintegerfn DDISABLE TERMINAL STREAM (integer STREAM, REASON)

This procedure is used to suspend (REASON=4) or abort (REASON=5) an input (STREAM=0) or output (STREAM=1) operation. For interactive I/O streams, only the "abort" option is likely to be required, and the effect is to send a "control message" through the communications system. For an output stream, as much unprinted data as possible are flushed from the system; for an input stream, similarly any data which have been typed are flushed from the system. In each case the stream is re-initialised and further operations may not be carried out on the stream until it has been re-enabled using the DENABLE TERMINAL STREAM procedure.

The DDISABLE TERMINAL STREAM procedure may be called at any time, but is typically invoked in response to an asynchronous "INT:" message generated by the user at the interactive terminal (see the procedure DCLEAR INT MESSAGE, described below, and the contingency-handling procedures, described in Section A1.6).

Possible error result: 8

****

externalintegerfn DCLEAR INT MESSAGE

When a terminal user presses the ESC key, the TCP software generates a prompt string "INT:". The message input by the user in response is called an "INT: message" and is transmitted to the user process independently of the input and output streams enabled by the DENABLE TERMINAL STREAM procedure. INT: messages may be at most 15 characters long, the character used to terminate the message not forming part of the message.

The EMAS 2900 System distinguishes single-character INT: messages from multi-character INT: messages (null INT: messages are discarded by the TCP). In the former case, the "contingency mechanism" is invoked; this is described in Section A1.6, below. The DCLEAR INT MESSAGE procedure has no relevance in this case.

Multi-character INT: messages, by contrast, are placed in a location visible to programs running at any level of privilege, and in particular to the subsystem. Thus an INT: message may arrive asynchronously with respect to the user process and may be used by it, for example, to re-direct the course of the computation or to take a particular action and then resume the computation.

The location of the INT: message may be referenced by using the address supplied in the AIOSTAT field of the UINF record (see Section 3.3 in the main text of the manual). The format of the record at this address is given by IOSTATF, below. Access to the location can be achieved as in the following example:

```
     .
     .
   recordformat IOSTATF(integer IAD, string(15) INT MESS, c
                        integer INBUFLEN, OUTBUFLEN, INSTREAM, OUTSTREAM)
   recordname IOSTAT (IOSTATF)
   stringname INT MESSAGE
     .
     .
   IOSTAT == RECORD(UINF_AIOSTAT)
   INT MESSAGE == IOSTAT_INT MESS~~AGE~~
     .
     .
   if INT MESSAGE = "STOP" start
     .
     .
```

When the process has noted the presence of a multi-character INT: message, the message should be cleared so that further examination of the message location will not yield the same message. The message can only be cleared by a call of the procedure DCLEAR INT MESSAGE, since the message location is not writable at the subsystem's level of privilege.

Any INT: message arriving before a previous message has been cleared overwrites the previous message.

Possible error results: none

externalintegerfn PRIME CONTINGENCY (routine ONTRAP)

The purpose of this procedure is to nominate a routine (the actual parameter) which is to be executed on the occurrence of various contingencies, both synchronous and asynchronous. The types of contingencies which will invoke the ONTRAP routine are as follows:

Synchronous

* Virtual store errors.

* Program errors.

* Instruction counter interrupt (treated as program error).

Asynchronous

* Arrival of single-character "INT:" messages (as in INT:A, for example, from an interactive terminal, or a TERMINATE to a batch process from the main Oper console).

* Arrival of text messages from the machine operator or from other processes.

* Arrival of "POFF" messages from other processes or from the Supervisor (available only to privileged processes).

The contingency scheme works as follows. If the subsystem has not called PRIME CONTINGENCY, and a contingency occurs, then the process is stopped with an Oper message SIGNAL FAIL 1. If PRIME CONTINGENCY has been called, the ONTRAP routine is executed, with LNB set to the value of SF at the time of the contingency, and SF set 7 words higher. (If there is deemed to be insufficient space above SF to execute the ONTRAP routine, LNB is set to a suitably lower value. In this case user diagnosis may be difficult. A main Oper message "OFF STACK TOP" is output. It is planned to treat this class later as a special contingency class.) The ONTRAP routine should be declared to have two integer parameters: CLASS and SUBCLASS. CLASS will be set (by Director when it calls the routine) according to the table on the next page.

SUBCLASS will normally be zero, but for single-character "INT:" messages it contains the "INT:" character.

Once PRIME CONTINGENCY has been called, the ONTRAP routine environment remains recorded, and is invoked on the specified contingencies until a further call of PRIME CONTINGENCY specifies a different ONTRAP routine or until a call of DRESET CONTINGENCY (see below) causes the first-set ONTRAP routine environment (if any) to be reinstated.

However, before the ONTRAP routine can be re-invoked for a further contingency after a contingency has occurred, the function DRESUME (see below) must be called. Indeed, occurrence of a further synchronous

Contingency	CLASS
Floating overflow	0
Floating underflow	1
Fixed overflow	2
Decimal overflow	3
Zero divide	4
Bound check	5
Size error	6
B overflow	7
Stack error	8
Privilege	9
Descriptor	10
String	11
Instruction	12
Accumulator	13
Virtual store error	32
Instruction counter	64
Single-character "INT:"	65
Text message	66

interrupt, other than "instruction counter", before DRESUME is called will cause repeated contingencies, culminating in a SIGNAL FAIL 3 (signal loop stop). Execution of the ONTRAP routine will normally be followed by a call of READ ID (read interrupt data, described below), which gives details of the process environment at the time of the contingency, and then a call of DRESUME.

If an asynchronous interrupt has occurred and the ONTRAP routine is entered, further asynchronous interrupts are inhibited until a call of DRESUME is made. Inhibited asynchronous interrupts are queued up to a current maximum of 4, after which they are discarded, with a main Oper message MESSAGE LOST.

Note that, at process start-up, asynchronous messages are inhibited until the subsystem calls DRESUME, LNB=-2. Also note that only asynchronous contingencies can be queued.

Possible error results: none

****

externalintegerfn READ ID (integer ADR)

This function provides the process environment as it was at the occurrence of the most recent contingency. The information is written to the area whose address is given by parameter ADR. Currently 18 words - the processor registers plus PC of erring or current instruction - are returned, but space should be left for 32 words altogether (for future extensions).

The interrupt data remains available for repeated reading until a call of DRESUME (described below) is executed. An error result of 53 is returned if no interrupt data is available, or of 45 if the address supplied is not accessible.

Possible error results: 45, 53

****

externalroutine DRESUME (integer LNB, PC, ADR18)

This routine is provided first to notify Director that the subsystem's ONTRAP routine (see the PRIME CONTINGENCY procedure, above) has completed that section of its code during execution of which it could not reasonably resume following a subsequent asynchronous contingency. Asynchronous contingencies are inhibited by Director until this notification is received; program or virtual store errors will cause looping.

Likely tasks in this inhibited state (called the "ONTRAP state") are:

i) determination of a new environment (e.g. an "ABORT" position, or a diagnostic routine), or

ii) output of a text message, or

iii) determination of other subsequent action (e.g. ignore asynchronous interrupt, and continue the former computation).

Secondly, DRESUME provides the means for the ONTRAP routine to specify the original or a chosen new environment, for resumption of normal computation.

The action taken is determined by the value of the parameter LNB. Parameter ADR18 is relevant only when LNB=0, and PC only when LNB is greater than zero.

LNB=-2	At process start-up, asynchronous interrupts are inhibited until the subsystem calls this entry point to DRESUME. This enables the subsystem to initialise itself properly for action. Subsequent calls of this entry point have no effect.

LNB=-1	This entry point tells Director that the ONTRAP routine has completed its critical operations and can again accept asynchronous interrupts. If a contingency is in fact queued at the time that this entry point is called by the ONTRAP routine, a new invocation of the ONTRAP routine is immediately created. The environment supplied, by procedure READ ID, to the new ONTRAP routine is that following the call of DRESUME in the previous ONTRAP routine.
LNB=0	This entry point may be called only from the ONTRAP state (otherwise the process stops with a main Oper message SIGNAL FAIL 10) following an asynchronous contingency or program error (SIGNAL FAIL 9 following a virtual store error). The call informs Director that computation is to be resumed at the point of interruption. Parameter ADR18 must point to a copy of the environment supplied to the subsystem by the READ ID procedure. The process leaves the ONTRAP state.
LNB>0	The values LNB, PC are transferred to the process registers LNB and PC, and computation continues at PC. The values assigned to the other registers are indeterminate; therefore it is expected that PC will point at or near an EXIT instruction so that a well-defined environment can be reached. The LNB value must be more than 5 words below the SF at the time of the contingency; otherwise SIGNAL FAIL 7 occurs. The process leaves the ONTRAP state.

Possible error results: none

****

externalintegerfn DISC ID

This function has exactly the same effect as calling DRESUME(-1,0,0). It should be called from the ONTRAP routine and uninhibits asynchronous contingencies.

Possible error results: none

****

externalintegerfn DASYNC INH (_integer_ ACT, ADR2)

This procedure supplies subsystems with a means of inhibiting asynchronous contingencies for short periods of time (such as that required for updating critical tables). A call with ACT=0 is used to initialise the arrangements for inhibiting contingencies; ADR2 must be the address of two words permanently available in read and write modes to Director (if the words are, or become, inaccessible, Director treats asynchronous contingencies as though this call had not been made).

The first of the two words is inspected by Director whenever an asynchronous contingency is received. The contingency is queued if the word is positive; otherwise it takes immediate effect. The second word is updated (if accessible) by Director whenever an asynchronous message is received and whenever one is un-queued; the word contains the number of messages currently queued, and may be inspected by the subsystem, for example when it wishes to uninhibit contingencies by setting the first word to zero. If the second word is positive it may then choose to invoke the queued contingency, by calling

```
DASYNC INH(1,0)
```

Possible error result: 8

****

externalroutine DRESET CONTINGENCY

This procedure causes the first-set ONTRAP routine to be reinstated as the contingency procedure.

Possible error results: none

A1.7 "Outward call" and subsystem "System call" procedures

externalintegerfn DNEW OUTWARD CALL (integer NEWACR, EMAS, c
NEW STACK SEG, DR0, DR1, integername I, J)

A System call table entry is created for an outward call, according to Ref. 8. In the entry created, KEY is set equal to the ACR of the program calling this function; E is set to 0 or 1 according to the value of EMAS; Target SSN is set with the value of NEW STACK SEG, which must be even and in the range 34 to 126. DR0 and DR1 should supply the descriptor descriptor to the required entry descriptor for the procedure to be outward-called. I and J are set on return to the i and j values to be used in making the outward call.

Only one of these System call table entries will be extant at any time - a further call of DNEW OUTWARD CALL will cause the previous outward call entry to be lost.

Possible error result

8 EMAS not 0 or 1.
New SSN not even and in range 34-126.
New ACR not greater than or equal to current ACR.

The descriptor supplied is not checked.

****

externalintegerfn DNOMINATE STACK (integer SEG)

This function is provided to satisfy requirement number 4 of Ref. 8, by calling the OUT12 (defined in Ref. 9) to notify the Local Controller to enable it to supply the locked-down SSN+1.

Possible error result

8 SEG not an even number in range 34-126. Segment SEG+1 is connected.

****

__externalintegerfn__ DNEW INWARD CALL (__integer__ ACCESS KEY, DR0, DR1, __c__
__integername__ I,J)

A new System call table entry is created for an inward call to the current ACR (i.e. the ACR of the program calling this function). ACCESS KEY specifies the ACR at and below which access is allowed to the specified procedure, and DR0, DR1 should be a descriptor descriptor to the entry descriptor for the procedure. The i and j values specifying the System call table entry are returned in I and J.

Entries created by this function are cumulative and cannot subsequently be removed from the System call table.

Possible error results

8 KEY ACR not greater than or equal to the caller's ACR

55 SCT full

The supplied descriptor descriptor is not checked for validity.

A1.8 Message passing procedures

externalintegerfn DPON2(string(6) USER, recordname P, c
integer MSGTYPE, OUTNO)

This function is used to pass System messages, as referred to in Ref. 10.

The format of the record P referred to in the parameter list is

(integer DEST, SRCE, P1, P2, P3, P4, P5, P6)

By "PON" we mean	send a message
By "POFF" we mean	take next message, or if none available suspend until one is available
By "TOFF" we mean	take next message if one available; otherwise set P_DEST=0 and continue execution

By "sync1", "sync2" and "async" service numbers, we mean the "N2", "N3" and "N4" service numbers referred to in Ref. 10.

No checking is performed by Director on the parameters passed to this procedure.

USER	is the 6-character username of a paged process to which the record P is to be sent. (Relevant only when the left-hand half of P_DEST=X'FFFF'; see Ref. 10.)
P	is a record containing the 32-byte System message to be dispatched. (The right-hand halves of DEST and SRCE are unchanged during the process of dispatching the message; the left-hand half of SRCE, and also of DEST when the left-hand half of DEST=X'FFFF', are set by the Local Controller.)
MSGTYPE	should be set to 1, 2 or 3 to indicate that the message generated is of the sync1, sync2 or async (i.e. N2, N3 or N4) type respectively. (Relevant only when the left-hand half of P_DEST=X'FFFF').
OUTNO	is the "OUT" number which is to be used, valid numbers being 5, 6, 7, 8, 9 or 10 (see Ref. 10).

The result of the function is always zero, except when the left-hand half of P_DEST=X'FFFF', when it is 61 if there is currently no process owned by the given username; the message has not then been dispatched. P_DEST may be set on return to indicate error conditions, as described in Ref. 10.

****

externalroutine DOUT (recordname P)

This is equivalent to DPON2("",P,0,7), causing the message P to be dispatched and the calling process to be suspended until a reply on the process's "sync2" service number is available; it is then received into the record P.

****

externalroutine DPON (recordname P)

This is equivalent to DPON2("",P,0,6), causing the message P to be dispatched and allowing the calling process to continue processing.

****

externalroutine DPOFF (recordname P)

This is equivalent to DPON2("",P,0,5) with P_DEST set to zero. N"o message is generated and the calling process is suspended until a message on the process's "sync1" service number is available; it is then received into the record P.

****

externalroutine DTOFF (recordname P)

This is equivalent to DPON2 ("",P,0,6) with P_DEST=0. No message is generated and the calling process always continues to execute. If no message on the process's "sync1" service number was available, P_DEST will still be zero; otherwise P contains the received message.

****

externalroutine DOUT11 (recordname P)

This invokes the Supervisor "OUT" no. 11, which causes the message P to be dispatched and the calling process's pages to remain in main store until a reply is available (not one of the sync1, sync2 or async replies referred to above, but one having the DEST field equal to the SRCE field of the outgoing message, the Local Controller having set the left-hand half of SRCE). The reply is received in the record P.

This "OUT" number should be invoked only when it is certain that a reply will be received, and in a very short time (e.g. the duration of a single magnetic tape transfer).

****

externalroutine DOUT18 (recordname P)

This invokes the Supervisor "OUT" no. 18, which causes the message P to be dispatched with the following side effects:

1. The local virtual store described by P_P5 and P_P6 (P5 = number of bytes, not exceeding (2**24)-1, P6 = address of start of area) is held in main store until a reply is available corresponding to the dispatched message.

2. Director places the caller's ACR value in bits 4-7 of P_P5 before executing the OUT instruction.

3. The Local Controller replaces the contents of P5 and P6 with the Local Segment Table base and limit for the process, and the ACR value, as required for the first two words of a GPC request block.

4. The caller may in addition set bit 0 of P_P5; in this case the Local Controller marks the page table entry (or entries) describing the store area as "written-to", thus ensuring that the pages are returned to disc when the process is removed from store, if required. (The GPC does not so mark page table entries into which it transfers data, for example.)

On return from this routine, the caller should check that P_DEST is not -1, which indicates either a failure to "lock down" the specified area of store or a condition preventing dispatch of the message.

****

```
externalintegerfn DSPOOL (string(6) USER, string(11) FILE, integer c
                          FSYS, QUEUE, COPIES, FILE, TYPE, FORM)
```

This procedure generates a spool request message to the Spooler process for file FILE belonging to user USER on disc FSYS. The remaining parameters define the Spooler actions, and the file will normally be transferred to Spooler's ownership.

The result of the function is 61 if the Spooler process is not available, or 0 if the request was successful. Other values represent error results from the Spooler process.

****

```
systemroutine OPER (integer OPERNO, string(255) S)
```

This procedure causes the text string S to be displayed on logical Oper console number OPERNO (taking as many lines of scrolled Oper output as necessary).

****

```
externalintegerfn DMESSAGE (string(6) USER, integername LEN, c
                            integer ACT, FSYS, ADR)
```

ACT=0 Delivers to ADR the next message sent to this user process. LEN should be previously set to the maximum length you are prepared to accept; it is set by the procedure to the number of bytes returned. LEN will be set to zero if no further messages are available.

ACT=1 Sends message (ADR, LEN) to USER on FSYS. Currently LEN is restricted to 200 bytes. Result 61 from this call means that a process belonging to USER is not currently present (but the message goes into his file anyway).

ACT=2 Determines whether USER on FSYS currently has a process. Result 61 if not, else zero.

ACTs 1 & 2 will in due course be subject to pseudo-permissions on the relevant message files, and to message-permission attributes in the SFI.

A1.9 Magnetic tape procedures

```
externalintegerfn DMAG CLAIM (string(6) TSN, integername SNO, c
                             integer REQ, MODE)
```

The magnetic tape volume labelled TSN is either claimed (REQ=0) or released (REQ=1). MODE should be set to 1 to allow read access to the tape, or to 2 if write access is required.

If a CLAIM call is successful, SNO is set with a number to be used in subsequent calls of DMAG IO.

If TSN is null in a RELEASE call, then all claimed tapes are released.

Possible error results: 8, 67

****

```
externalintegerfn DMAG IO (integername REPLY FLAG, CONTROL, LEN, c
                          integer TYPE, SNO, ADR)
```

The parameter TYPE determines the action of the procedure:

TYPE =	0	erase
	1	read
	2	write
	3	check-write (not yet implemented)
	4	check-read (no data transfer)
	5	private chain (not yet implemented)
	6	rewind to BT
	7	spare
	9 ~~8~~	tape position
	8 ~~9~~	file position
	10	write tape-mark

SNO should be set with the value returned by a previous CLAIM (procedure DMAG CLAIM). ADR and LEN specify the area from or to which data is to be transferred (if relevant). In the case of TYPE=1 (read), LEN is set on return to be the number of bytes transferred.

CONTROL is used to specify detailed actions by the tape handler routine for certain requests, and should normally be zero. It may be set on return to indicate the occurrence of certain conditions during execution of a request. (A specification for the use of this parameter will be issued when necessary.)

REPLY FLAG is set to zero for a successful operation, or

1	failure (parity etc.)
2	request rejected
4	beginning of tape, end of tape or unexpected tape mark found, but operation otherwise successful

TYPE=~~9~~ 8

For request TYPE=9 (file position), LEN specifies the number of blocks to be skipped: negative means backwards, positive forwards; zero is invalid. If a tape-mark is found (REPLY FLAG=4) then a skip back of one block is performed before the reply is given. It is thus impossible to pass a tape-mark except by the use of a TYPE=8 (tape position) request.

TYPE=~~8~~ 9

For request TYPE=8 (tape position), LEN is the number of (notional) files to be skipped, and CONTROL is the number of tape-marks in a notional file; CONTROL must be positive. If LEN is negative the skip is backwards, if positive then it is forwards; zero is invalid. CONTROL*LEN tape-marks are skipped. A backwards skip will stop at BT. A forward skip will stop if the first block within a (notional) file is a tape-mark; the tape will then be positioned before the tape-mark and REPLY FLAG=4 will be returned. Thus if CONTROL=1 and LEN="very large", the tape will be positioned between the first double tape-mark encountered.

A1.10 Error messages used by Director procedures

0		SUCCESSFUL
1	PRG/UNPRG	FILE>1SEG, SITE NOT 40-ALIGNED
2	PRG	DISC NOT AVAILABLE
3	TRANSFER	DIFF SECTSI
4	TRANSFER	FSYS1/2 NOT MATCH
5	CONN/DESTR	FILE NOT READY
6	CONNECT	FILE ON OFFER
7	INDEX CREATE	NO INDEX SPACE
8	VARIOUS	BAD PARAM
9	ACCEPT	NOT ON OFFER
10	ALLOCATE	FSYS FULL
11	CRE/REN/ACC	NOT VALID FOR THIS USER
12	INDEX CREATE	REQ SIZE NOT 4<=NKB<=32
13	INDEX CREATE	NNT FULL
14	INDEX CREATE	USER ALREADY HAS INDEX
15	CREATE	NO FREE FDS
16	CREATE	ALREADY EXISTS
17	CREATE	INSUFFICIENT FREE CELLS
18	CRE/DESTR/FINF	BAD FILENAME
19	ALLOCATE	10 DISC CLEARS FAILED
20	OFFER	ALREADY ON OFFER
21	DESTROY	INTERNAL ERROR
22	DEAL	BITS ALREADY CLR FOR SOME PGS
23	GETUSNAME(S)	FSYS NOT AVAILABLE
24	DESTROY	USER NOT KNOWN
25	DESTROY/TRANSF	FILE IN USE
26	CONNECT	INVALID SEG, LEN OR APF
27	PRG/UNPRG	FILE TOO BIG
28	SCONNECT	CBT FREELIST EMPTY
29	CONNECT	BAD FD - NO PAGES
30	CONNECT	INVALID FILE, MODE, APF
31	CCKMESS	INVALID HDR
32	VARIOUS	NOT EXIST / NO ACCESS
33	CONNECT	CONFLICTING MODE
34	CONNECT	ALREADY CONNECTED THIS VM
35	CONNECT	SEG IN USE OR GAP TOO SMALL
36	NOM/DE-NOM	FAILS
37	GET USN	USER NOT KNOWN
38	CREATE	MAX KB FOR USER
39	DISCONNECT	NOT CONNECTED
40	RENAME	FILE CONNECTED
41	CHFISI/CREATE	INVALID NEW SIZE
42	CHSIZE	CONN IN ANOTHER VM
43	CHSIZE	NOT ENOUGH FREE CELLS
44	CHSIZE	READ OR WRITE FAILURE
45	VARIOUS	USER PARAM AREA NOT ACCESSIBLE
46	DPERMISSION	BAD PERMISSION OR USERNAME
47	SYSTEM CALL	NOT ENOUGH STACK
48	SYSTEM CALL	NOT ENOUGH INSTRUCTIONS
49	DPERMISSION	MAX (16) ITEMS ON PRM LIST
50	DPERMISSION	USER NOT IN LIST
51	DESTROY	OWNP IS ZERO
52	DFSTATUS	FILE CONN WR-MOD
53	READID/DISCID	NO INTERRUPT DATA
54	NEW IN CALL	NO OUTCALL SET UP
55	NEW SCT ENTRY	SCTABLE FULL
56	LOCK	PAGE NOT CONN OR INVALID

57	RETURNPAGE	PAGE NOT ALLOC
58	EXEC CHAIN	INVALID PARAM / RCB NOT ACCESSIBLE
59	ARCHIVE	CHECKSUM FAIL
60	VARIOUS	INVALID PARAMS
61	INTER-PROCESS	PROCESS N/A
62	BAD PAGE	NOT IN LIST
63	BAD PAGE	BAD BAD-PG-COUNT
64	BAD PAGE	LIST FULL
65	CCK	BAD DT
66	AUTO-CLOSE	CLOSE SEQUENCE CANCELLED
67	MAG TAPE	TAPE (NOT) CLAIMED
68	MAG IO	LCK DOWN FAILS
69	CCK	ALREADY DONE
70	XOPER OPEN TO	USERLIST FULL
71	STARTP	SYSTEM FULL
72	STARTP	CANNOT START PROCESS
73	MAG TAPE	MAX TAPES ALLOCATED
74	STARTP	PROCESS RUNNING
75	ARCHIVE	INDEX CORRUPT
76	STARTP	WORK FILE FAILURE
77	ARCHIVE	INDEX IN USE
78	DDISCONNECT	INDEX ERROR (GFIND)
79	MAP INDEX	INDEX CORRUPT
80	DMONENABLE	>1 SECTION

__externalintegerfn__ DPLUGIN JVECTOR (__integername__ IVALU, __c__
   __integer__ ADDRESS, TOP ENTRY)

ADDRESS points to the zeroth element of a recordarray (0:TOP ENTRY), format (__integer__ TYPE, ACR, DR0, DR1), which is to be a new ("vertical") J-vector in the system call table. If call is successful, IVALUE is set to the I-value to be used in subsequent system calls accessing the vector. The procedure allows a subsystem to cause outward calls and to allow users programs to do inward calls to the subsystem. The address must lie in a non-write-access file, readable at Director's ACR No target ACR in any of the entries must be less than the caller's ACR , nor may the target PRIV bit be set. After a successful call, the file may not be disconnected (except by calling DSTOP), nor may its size or access mode be changed (flag 84 if attempt is made). The vector may contain a maximum of 512 system call entries (else result 8). Subsequent calls of DPLUGIN JVECTOR cause the previously plugged-in vector to be forgotten (though the file still may not be disconnected).

.ossible error results: 8,45.

__externalintegerfn__ DLOWER ACR (__integer__ NEWACR)

This privileged procedure (currently accessible from ACR 4) enables the calling procedure to acquire a lower ACR (>0) and optionally to acquire PRIV (bit 2**4 set in parameter NEWACR) and/or DGW and ISR in SSR (bit 2**5 set in NEWACR).

Possible error results: 8   New ACR not between 1 and caller's ACR

APPENDIX 2

FINDING OUT ABOUT SPOOLER'S CONFIGURATION

Contents

A2.1 Introduction

As explained in Reference 2, the paged System process Spooler (process name SPOOLR) maintains a file index on each of the on-line file system devices, and in these indexes keeps queues of files to be output to local peripherals and remote job entry (RJE) stations, and queues of batch jobs to be run. Spooler also handles all local card reader and paper tape input and RJE input.

This appendix explains how a subsystem can determine which queues are currently available to it. It was originally published as EMAS 2900 Spooler Note 2, and was written by Bill Laing.

The subsystem must first connect a file belonging to Spooler called SPOOLR.CFILE in read shared mode, with the file system number specified as -1. File CFILE has a standard Edinburgh Subsystem header, with the last two type-dependent words specifing the number of Spooler's queues and the number of Spooler's streams. The format of the file header is defined below.

Spooler has this file connected in write mode with read access permitted to all users. Therefore it should be borne in mind that some of the values in this file are constantly changing. Two IMP record arrays are maintained in this file, one describing the queues in the spooling system and the other the streams managed by the spooling system.

A2.2 Record formats and sizes

A2.2.1 File header format

Size 32 bytes.

```
recordformat FHF(integer END, START, SIZE, TYPE, SPARE, c
                         DATETIME, QUEUES, STREAMS)
```

START	specifies the number of bytes from the start of the file at which the record array describing the queues starts. The record array describing the STREAMS starts after the queues array, i.e. (START + QUEUES * queue entry size) bytes from the start of the file. The queue entry size is currently 128 bytes (see A2.2.2 below).
QUEUES	specifies the number of elements in the record array describing the queues in the configuration.
STREAMS	specifies the size of the record array describing the streams in the configuration.

A2.2.2 Queue entry format

Size 128 bytes.

```
recordformat QUEUEF(string(15) NAME, c
   byteintegerarray STREAMS(0 : 15), c
   string(7) DEFAULT USER, c
   string(31) DEFAULT DELIVERY, c
   integer DEFAULT START, DEFAULT PRIORITY, DEFAULT TIME, c
       DEFAULT FORMS, DEFAULT MODE, DEFAULT COPIES, DEFAULT RERUN, c
       LENGTH, HEAD, MAX LENGTH, MAX ACR, SPARE2, SPARE3, AMOUNT)
```

NAME	specifies the name by which the queue is known.
STREAMS	specifies the streams which are serving this queue. If an element in the array is non zero it contains a stream number i.e. an index into the stream record array.
DEFAULTs	various default values which a document is given when added to the queue, if they are not already specified.
LENGTH	the current number of entries in the queue.
MAX LENGTH	the maximum length to which the queue is allowed to grow.
MAX ACR	the maximum ACR level at which a user process can be and still be able to access the queue.
AMOUNT	the total amount, of bytes or seconds as appropriate, currently queued. If DEFAULT TIME <= 0 then the amount is in bytes; otherwise the amount is in seconds.

A2.2.3 Stream entry format

Size 80 bytes.

```
recordformat STREAMF(string(15) RMT, DEVICE, c
   integer STATUS, NAME, QUEUE, LIMIT, c
   byteinteger FORMS, LOWEST PRIORITY, HEADER TYPE, c
   HEADER NUMBER, integer SPARE1, COMMS STREAM, SECTION, c
   BYTES SENT, BYTES TO GO, SERV ROUT, DOCUMENT)
```

RMT	the location of the device or batch stream.
DEVICE	the name of the stream.
STATUS	the current status of the stream: 0 - CLOSED 1 - OPEN 2 - IDLE 3 - ACTIVE 4 - CONNECTING 5 - DISCONNECTING 6 - ABORTING 7 - SUSPENDING
NAME	is only relevant for input or output streams. NAME is derived from three numbers: 1. DEVICE TYPE<<24 2. DEVICE NUMBER<<16 3. EXTERNAL STREAM NUMBER, if DEVICE TYPE is a Front End Processor Device Types: 3 - Card Punch 4 - Card Reader 6 - Line Printer 14 - Front End Processor
QUEUE	the queue number which this stream is attached to, unless it is an input device, when the queue number specifies the output queue for that input device.
LIMIT	the current limit on that stream, i.e. in Kilobytes or Seconds.
FORMS	the current forms type selected.
LOWEST PRIORITY	the lowest priority of document which the stream will currently handle.
HEADER TYPE	the type of banner generated if the stream is an output device: 0 - NONE 1 - LINE PRINTER 2 - TAPE PUNCH 3 - CARD PUNCH

COMMS STREAM the communications stream currently allocated to the stream if the stream is an input or output device.

SERV ROUT specifies the type of stream:

0 - Output Device
1 - Input Device
2 - Job, i.e. batch job
3 - Process, i.e. JOBBER or JOURNL

DOCUMENT if non-zero, specifies the current document being handled by the stream. It is interpreted in the form FSYS<<24!DOC NO, expressed as a six-character string "nndddd", with "nn" in the range 00 - 99 and "dddd" in the range 0001 - 9999.

INDEX

Edinburgh Subsystem Information

Information for: %SYSTEMROUTINE REROUTECONTINGENCY   Last updated: 06/06/79

Parameters:
%INTEGER EP,CLASS,%LONGINTEGER MASK,%ROUTINENAME RR,%INTEGERNAME FLAG

Use:
Used to re-route selected contingencies to a nominated routine RR. The nominated routine must take two parameters of type %INTEGER, CLASS and SUBCLASS. EP can take the following values:

EP	Meaning
0	Clear all re-routeing
1	Re-route all occurrences of CLASS
2	Re-route occurrences of CLASS if bit 2**SUBCLASS is set in MASK (deals with SUBCLASS 0-63).
3	Re-route occurrences of CLASS if bit 2**(SUBCLASS-64) is set in MASK (deals with SUBCLASS 64-127).
4	Re-route all classes specified as bits 2**CLASS in mask (deals with classes 0-63)
5	Re-route all classes specified as bits(2**(CLASS-64) in MASK (deals with classes 64-127)

Note that only a limited number of re-route requests can be in effect at any time (currently 8). However by use of the mask facility it is possible to re-route groups of contingency classes or groups of subclasses within a single Class.
Note that no consistency check is made between calls. If the same CLASS-SUBCLASS pair is re-routed twice the effect is not defined.
Note that this routine is called with EP=0 at the start of each command by the basic command interpreter.
Faults are returned via FLAG. They are 202 (invalid parameter) and 300 (table full).

Status:
Stable apart from backward compatible extensions - No Advisory support