EMAS 2900

Notes

Benchmark

BENCHMARK

EMAS 2900 [redacted] NOTE

No: 1 1
Date: 4/10/78

Facilities for Running a Test of Many Users without the Interactive Network

This note describes a mechanism incorporated in the Edinburgh Subsystem and utilising a program in MANAGR to run many processes. It does not use ERTE (the remote terminal emulator); it does not even require an interactive network. It is useful for running tests on mainframes without interactive networks, and for running hardware or software confidence tests.

The method is to create a number of user processes with the generic name "TEST" and to start these. This particular name is detected by the Subsystem and instead of requesting input from the interactive terminal it executes the following commands:

```
OBEY(MANAGR.TESTOBEY,.LP)
STOP
```

The file MANAGR.TESTOBEY can contain any sequence of commands - e.g. compilation of a file and running a program. It is important that MANAGR.TESTOBEY, and any other files required, be permitted to the TEST process.

To simplify the creation and starting up of the user processes the following command is available in MANAGR:

```
CREATETESTUSERS
```

This prompts for file system, first user number and number of users. Thus if one wanted to create 10 users starting from TEST01 on file system 2, the replies would be as follows:

```
FSYS: 2
FROM "TEST" NO:1
HOW MANY? 10
```

There is an equivalent command DELETETESTUSERS.

The command STARTTESTUSERS is used to start some of the TESTUSERS. It prompts for the first user and number of users, and then sends start requests to the process "DIRECT".

R.R. McLeod

EMAS 2900 BENCHMARK NOTE

No: 2
Date: 14/11/78

Benchmark Utilities

This note describes some utilities involved in the initialisation and use of the F1 benchmark. It is intended as a record rather than a working document, and anyone wishing to make use of any of the utilities described should first contact the author.

The routines are contained as separate object-file members of ERCC28.BENCH.

1. BECKON(from,to)

   This routine causes all the files in an ECSB process on the 4-75 to be copied into a pdfile (of the same name as the process) and then transferred to process ERCC28 on the 2970.

   The parameter defines the range of ECSB processes to be operated upon.

2. UPDATEPD(pdfile)

   This routine overwrites pd members in files named ECSBnn with similarly named members in file "pdfile"; thus new versions of files can be placed in the pdfiles corresponding to each benchmark process.

   The reply to the prompt "process nos:" is two integers defining the range of ECSB pdfiles to be updated.

3. MAKEVCON(file)

   This routine creates a character file which can be OBEYed to write a Volumes TRANSFER tape to create ECSxnn processes, where x and nn are specified as ranges of values. x can be B, E, F or G.

   A prompt is given asking for the range of processes for which the tape is to be written; e.g. 11 48 indicates processes ECSx11 - ECSx48.

   The next prompt asks for the number of sets of processes to be created:

   The reply 1 will create entries for processes ECSBnn
   The reply 2 will create entries for processes ECSBnn and ECSEnn
   The reply 3 will create entries for processes ECSBnn, ECSEnn and ECSFnn
   The reply 4 will create entries for processes ECSBnn, ECSEnn, ECSFnn and ECSGnn

   The vol label of the tape to be written is then prompted for.

   The tape is written to by issuing the command OBEY(file) and can then be used to create (or overwrite) files in ECSx processes by the OPER command V/TRANSFER <vol label>

The remaining routines must be run in MANAGR and are used to manage F1 processes.

4. CREATE F1 PROCESSES

Command:INSERT(ERCC28.BENCH_F1PROCY) may be required.

The following prompts are issued:

process range:	p1 p2	where 0 <= p1 <= p2 <= 99 (corresponding to ECSx00-99)
set range:	s1 s2	where 1 <= s1 <= s2 <= 4 (corresponding to ECSB,ECSE,ECSF,ECSG)
no of disks:	d	where 1 <= d <= 16
fsys nos:	list	where 0 <= list item <= 99

The program relays NEWUSER commands to DIRECT to create the processes. Successive user numbers are generated with process number within set number, e.g. ECSB11, ECSB12, ECSB13, ECSB14..., ECSE11, ECSE12, ECSE13 ...., etc. and assigned cyclicly to the nominated disks.

5. DELETE F1 PROCESSES

This issues prompts for process range and set range. In this case, DELUSER commands are relayed to DIRECT to delete the processes. Each process is deleted from the first disk pack on which it is found.

6. START F1 PROCESSES

Prompts are issued for process range and set range. The program then relays START commands to DIRECT for each process.

A. Shaw

Setting up Processes for the F1 Benchmark

This note describes a procedure for creating and initialising the EMAS 2900 processes required for running the F1 benchmark.

Two magnetic tapes are required:

F1PROC, which contains all the files required by the F1 processes.

F1UTIL, which contains the contents of process ERCC28; this comprises the following:
- a pdfile for each F1 process containing all the files required by the process.
- a group of utility programs in file BENCH.

Proceed as follows:

1. Create process ERCC28 by the OPER command:

```
D/NEWUSER ERCC28 n 4      where n is the disk-pack number
```

2. Load ERCC28 from tape. First mount F1UTIL, then issue the OPER Command:

```
V/TRANSFER F1UTIL
```

3. Log on to ERCC28 and perform various initialisations:

```
Command:INSERT(BENCH_UPDATY)
Command:INSERT(BENCH_MAKEY)
Command:INSERT(BENCH_MTSUPPY)
Command:INSERT(BENCH_VTAPE01Y)
Command:PERMIT(BENCH,MANAGR)
```

4. Create the benchmark processes:

```
Log on to MANAGR
Command:INSERT(ERCC28.BENCH_F1PROCY)
Command:CREATEF1PROCESSES
PROCESS RANGE:11 48
SET RANGE:4
NO OF DISKS:      [reply as
FSYS NOS:          appropriate]
```

5. Load the processes with their files; first mount F1PROC, then give the OPER command:

```
V/TRANSFER F1PROC
```

6. The processes are now ready for use.

Three other utility programs, held as members of pd file ERCC28.BENCH, may be of interest:

a) DELETE F1 PROCESSES

To redistribute the F1 processes or change the number of disks used, first delete them using this program. Then repeat from the middle of step 4 onwards.

b) Two other programs are used to modify the files used by the F1 processes. Rather than amending files in the processes themselves, it is preferable to amend the copies held in the pdfiles in ERCC28. Since some files are used by more than one process, UPDATEPD is provided to ensure that all copies of a file (held in various pdfiles) are updated at once.

The method is to gather all amended files into a new pdfile, then call UPDATEPD with this pdfile as parameter.

The next step is to run MAKEVCON which generates a file containing the information required to write a new F1PROC tape. The tape is written by OBEYing the file produced.

The OPER command V/TRANSFER <tape> completes the operation.

The benchmark utility programs are described in Benchmark note no. 2.

A. Shaw

Director

No: 2
Date: 26/4/77

The File System for EMAS 2900

1. Introduction

This note is intended to give enough information about the file system to form an initial operational guide, but does not comprise a detailed description. The file system closely follows the philosophy of that for System 4 EMAS (see Rees: "The EMAS Director", Computer Journal Vol 18, no 2). 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 ~~the~~ following primitives, to be called by outer code (higher ACR, less privileged) - in particular the "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 supervisor (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 virtual addresses so 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 "user").

2. File space allocation and index maintenance

The basic unit of store in the virtual memory is the "epage" (E-page), currently 4 Kbytes but possibly subject to change. DIRECTOR organises the file system in terms of these epage units. A 100 Mbyte disc pack contains 24000 epages (approx. X'5E00'), and currently DIRECTOR uses from X'2000' to X'5000' for the file system. (Epages below X'2000' are used for the IPL supervisor, 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 epage per file index) 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 means that the epage is allocated. The bitmap is located at X'2000', the indexes start at X'2020', and the users' file pages start at X'2080'. These 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 256 Kbyte (or 1 segment-sized) file therefore has 4 sections. The index records the string name of the file, and a file descriptor in the index contains a chain of 4 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 number of epages (sections) 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 "secondary segment" and "claimed block" tables. The addresses of the starts of these tables are passed to DIRECTOR at process start-up.

Users' indexes are numbered according to the epage number at which they start; currently each index comprises one epage. A file system request from user or "Subsystem" specifies the file owner (6 characters) and the file name (up to 15 characters). In order to reference the file index for the owner, DIRECTOR first connects it into the virtual memory. To do this, it 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, and DIRECTOR connects that section into the virtual memory.

N.B. DIRECTOR connects bitmaps, name-number tables and indexes "on demand" into segments numbered between 16 and 31. For efficiency, the bitmap and name-number table for the pack on which the process owner's files and index reside, and also the section containing his index, are left connected for the duration of the process.

3. File System creation and maintenance

Process number 1 currently takes the role of creating and maintaining the file system. Other processes are considered "normal processes" and the action of DIRECTOR on start-up is to read from the process owner's index the identifier of his "BASEFILE" (default SUBSYS.BASEFILE), to create contingency stack and BASEGLA files, and to CONNECT all these. It then enters (jumps to) the BASEFILE, having raised the ACR level to "Subsystem" ACR number (and having collapsed the stack), using the EXIT instruction.

Process number 1 awaits messages from an OPER console. These are typed as follows:

press "COMMAND"
type "1/message"
press "ENTER"

"message" is then passed to process 1.

N.B. The last two characters (assumed to be decimal digits) of each EMAS disc label (assumed to be 6 characters) are taken to be the (decimal) number of the disc pack (loosely, the file system number) for file system purposes. For example, if the disc label is EMAS15 then the pack is said to contain "file system 15" (considering the whole file system as a set of file systems, each of which resides on one disc pack).

The available messages are as follows:

1/CLEAR FSYS n

This must be given prior to creating or re-creating a file system. The bitmap, name-number table and file index areas are cleared to zero. The bitmap semaphore word (SW) is set to -1. n specifies the file system (disc pack).

1/CCK n

(Consistency check for file system n)
This is used to prove the self-consistency of the file system and the validity of the indexes within it. It also re-initialises the semaphore words in the indexes. The command should always be given, assuming file system n already exists, after the system is started and before any other file system activity is invoked (e.g. starting another process). If n is -1 or is omitted, a consistency check is performed for each EMAS disc currently on-line. On completion the fullness of the file system is reported on the OPER console.

1/NEWUSER xxxxxx n

This is used to create an index and a name-number table entry for user xxxxxx (six alphanumeric characters) on disc pack n.

1/DELUSER xxxxxx n

This deletes the index and name-number table entry for user xxxxxx on disc pack n.

1/CIND xxxxxx n

This re-initialises (clears) the index for user xxxxxx on disc pack n.

1/DUMPI xxxxxx n

This prints, on the MAIN line printer, the index for user xxxxxx on disc pack n. The index appears as a hexadecimal dump which is partially annotated.

1/VALIND xxxxxx n

This validates the index for user xxxxxx on disc pack n, printing error messages on the MAIN line printer where internal errors are found. VALIND is also invoked by DUMPI.

1/BITMAP n

This prints the bitmap for disc pack n on the MAIN line printer (hexadecimal dump).

1/USERNAMES n.

This lists the users having indexes on disc pack n, with the starting epage number and number of epages for each index.

1/COLLECTG xxxxxx n

This performs a "garbage collect" on the index for user xxxxxx on disc pack n. This function will normally be invoked automatically, but currently needs to be done when File System Error number 15 occurs (e.g. in response to a CREATE request) for the user.

1/ERRORCODES

This prints the file system error codes on the MAIN line printer.

1/PRG
1/UNPRG

(Etymology uncertain)
This is provided for systems programmers to move copies of Supervisors, DIRECTORs etc. to and from the fixed disc sites from and to the file system.

'FILE FSYS'

is printed on the OPER, to which you respond

1/file n

e.g. 1/ERCC10.DIRFILEZ 1 (i.e. from or to disc pack 1)

'LABEL SITE'

is then printed on the OPER, to which you respond

1/label site

where 'label' is the disc label and 'site' is the start-page of the site allocated for the Supervisor or DIRECTOR code.

e.g. 1/EMAS01 X2C0

The contents of the file are moved to the fixed site, or vice versa. See also the document "Fixed sites on EMAS discs".

The remaining OPER input messages to process 1, seen on the listing of file XOPER, are mainly for (transient) testing purposes.

J.K. Yarwood

EMAS 2900 DIRECTOR NOTE

No: 4
Date: 24/11/78

The Director Interface: File System Procedures

The services provided by Director fall into the following categories:

1. File system creation, maintenance and monitoring (facilities to be used by the System Manager). Described in Director Note number 2.

2. File system procedures (to be used by Subsystems). DESCRIBED IN THIS NOTE.

3. Contingency-handling procedures (to be used by Subsystems).

4. Console i/o procedures (to be used by Subsystems).

5. Process metering and accounting procedures (to be used by Subsystems and by the System Manager).

6. Procedures for file or job output to local or remote devices or processors (to be used by Subsystems).

Each procedure is reached by system call. Definitions of the relevant I, J values and how to access them are the subjects of a separate document.

A list of error messages is given on the last two pages of this Note.

Conventions

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

2. An Epage is an EMAS logical page, currently 4 kbytes. (The Epage size is said to be 4.)

3. Each user has a file index, identified by the user name, comprising six alphanumeric characters.

4. A Filename, unique in an index, may be up to 11 alphanumeric characters and may also include the characters '#' or '.' (or both) except in the first position.

5. Since the procedures available as the EMAS 2900 Director interface do not supply facilities for program loading and 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 collection of facilities is referred to as a Subsystem.

6. A Subsystem may choose to run "user" computations at an ACR level higher than its own, thereby effectively preventing access to some of the Director facilities, other than via the Subsystem. For convenience in description, we assume here that all Director procedures are invoked by a Subsystem.

General Notes

which apply where explicitly indicated in the text.

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.

Routines described in this note

routine	page
DCREATE	4
DDESTROY	4
DCONNECT	5
DDISCONNECT	6
DCHSIZE	6
DRENAME	7
DOFFER	7
DACCEPT	7
DPERMISSION	8
DFINFO	10
DFSTATUS	10
DSFI	12
DFSYS	15
DNEWGEN	15
DFILENAMES	16
DTRANSFER	16
DGETDA	17
GET USNAMES	17
GET AV FSYS	17
DPRG	18
DUNPRG	18
ACREATE	18
DMOD ARCH	19
DRESTORE	19
DNEW ARCH INDEX	19

externalintegerfn DCREATE (string(6) USER, string(11) FILE, c
integer 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 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

****

externalintegerfn DDESTROY (string(6) USER, string(11) FILE, c
string(8) DATE, integer 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; it 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 c
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:

(Continued on next page)

1	4	4
E X	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 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

****

externalintegerfn DDISCONNECT (string(6) USER, string(11) FILE, c
integer 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 the file 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

****

externalintegerfn DCHSIZE (string(6) USER, string(11) FILE, c
integer 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

or, 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.

(Continued on next page)

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?? (indicating 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 as a whole, 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 VM
USE	the current number of users of the file
ARCH	the value of the archive byte for the file (see procedure DFSTATUS)
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 at the end of this Note)
CODES2	information for internal use (see the diagram at the end of this Note)
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.

Note 1 applies.
Note 2 applies.

(Continued on next page)

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 by called only be 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 result ⁻2 is given.
12	ARCH	Operation 2 (PRIVILEGED). Clear currently-being-backed-up bit (2**1) and has-been-connected-in-write-mode (2**0) bit.
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.

(Continued on next page)

15	CLR USE	Clear file use-count and WRITE-CONNECTED status (PRIVILEGED).
16	CLR NOARCH	Clear archive-inhibit bit in CODES.
17	SET NOARCH	Set archive-inhibit bit in CODES.

Note: 16 and 17 are for system libraries etc, and are both PRIVILEGED.

18	SSBYTE	Set SSBYTE to be bottom 8 bits of VALUE (byte for 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 Identifier)

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. 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)

(Continued on next page)

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	12xinteger
5	Foreground and background passwords (reading is a privileged operation); a zero value means "do not change"	2xinteger
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)	2xinteger
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 users, 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 Director Note 11) 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	Local process logfile	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.

Possible 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, c
  string(6) TAPE, 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 into the ownership of user USER2 on disc-pack FSYS2 under name NEWNAME. TYPE should be set 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 of the file

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 disc 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

****

externalintegerfn DUNPRG (string (6) USER, string(11) FILE, c
integer FSYS, string(6) LABEL, integer 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 disc labelled LABEL.

Note 2 applies.

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

****

externalintegerfn ACREATE (string(6) USER, TAPE, string(8) DATE, c
string(11) FILE, integer 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

****

externalintegerfn DNEW ARCH INDEX(string(6) USER, integer 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

****

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

This procedure passes a restore request (if FILE exists on archive storage and is permitted to the caller) to the Volumes executive. 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

****

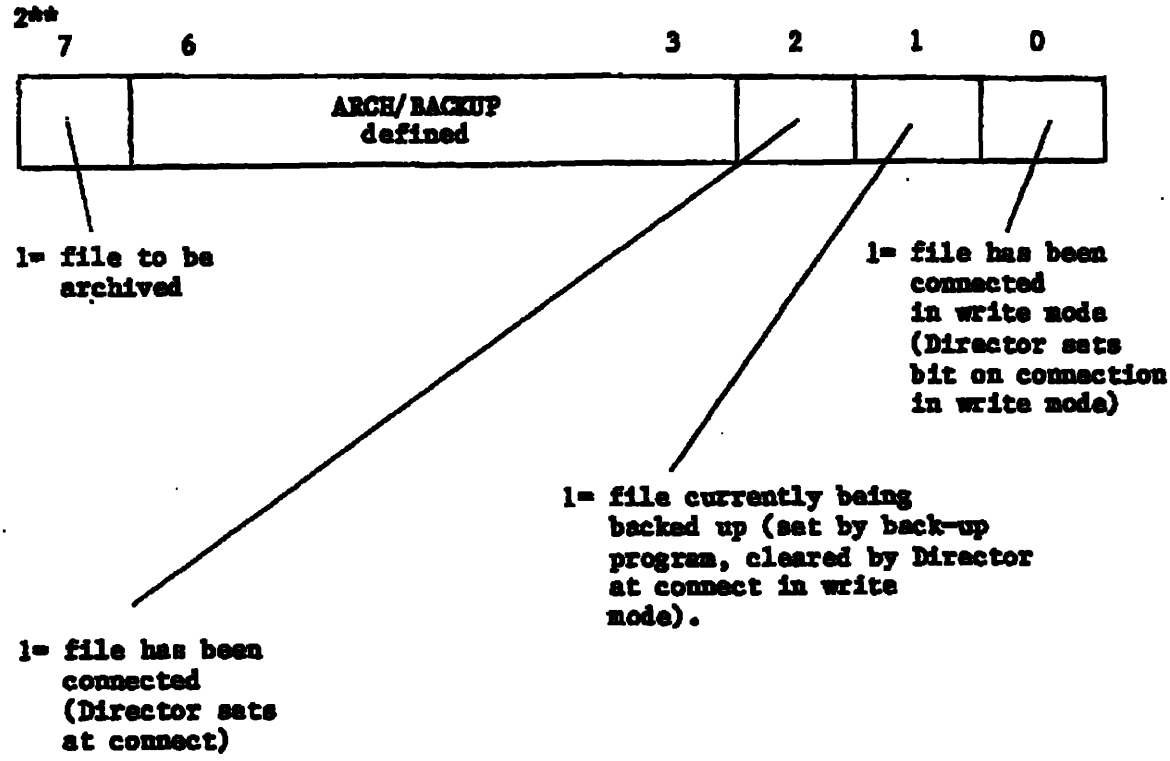
externalintegerfn DMODARCH(string(6) USER, string(11) FILE, c
string(8) DATE, recordname ENT, integer FSYS)

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

****

Contents of ARCHIVE byte



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

Contents of CODES2 byte

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

File System Error Codes

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

J.K. Yarwood

EMAS 2900 DIRECTOR NOTE

No: 5
Date: 11/7/78

The Director Interface: Contingency-Handling Procedures

The services provided by Director fall into the following categories:

* File system creation, maintenance and monitoring (facilities to be used by the System Manager). Described in Director Note number 7.

* File system procedures (to be used by Subsystems). Described in Director Note number 4.

* Contingency-handling procedures (to be used by Subsystems). DESCRIBED IN THIS NOTE.

* Interactive terminal I/O procedures (to be used by Subsystems). Described in Director Note number 13.

* Procedures for the implementation of outward call and Subsystem system calls. Described in Director Note number 6.

* Procedures for the implementation of message passing to other EMAS processes. Described in Director Number 11.

* Procedures for handling magnetic tape. Described in Director Note 12.

Each procedure is reached by System Call. Definitions of the relevant I, J values, and how to access them, are the subjects of a separate document.

A list of error messages is given on the last two pages of this Note.

Conventions

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

2. An Epage is an EMAS logical page, currently 4 kbytes. (The Epage size is said to be 4.)

3. Each user has a file index, identified by the user name, comprising six alphanumeric characters.

4. A Filename, unique in an index, may be up to 11 alphanumeric characters and may also include the characters '#' or '.' (or both) except in the first position.

5. Since the procedures available as the EMAS 2900 Director interface do not supply facilities for program loading and 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 collection of facilities is referred to as a Subsystem.

Routines described in this Note

routine	page
PRIME CONTINGENCY	2
READ ID	4
DRESUME	4
DISC ID	5
DASYNC INH	6
DRESET CONTINGENCY	6

---

externalintegerfn PRIME CONTINGENCY (routine ONTRAP)

The purpose of this procedure is to nominate a routine (the actual parameter in the call of PRIME CONTINGENCY) 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 later planned to treat this

case as a special contingency class.) The ONTRAP routine should be declared to have two integer parameters: CLASS and SUBCLASS. CLASS will be set according to the following table:

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

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 to be reinstated (if any).

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 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 of DRESUME.

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

Note that at process start-up, asynchronous messages are inhibited until the Subsystem calls DRESUME, entry LNB=-2 (q.v.).

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 at 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 job to which it could not reasonably resume following a subsequent asynchronous contingency. Asynchronous contingencies are inhibited 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 ADR1B is relevant only when LNB=0, and PC only for values of LNB 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 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 ADR1B 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 errors 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

****

File System Error Codes

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	CCKNESS	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	NON/BE-NON	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

J.K. Yarwood

The DIRECTOR Interface: Procedures for Implementation of Outward Call and Subsystem System Calls

The services provided by DIRECTOR fall into the following categories:

* File system creation, maintenance and monitoring (facilities to be used by the system manager). Described in DIRECTOR Note number 7.

* File system procedures (to be used by Subsystems). Described in DIRECTOR Note number 4.

* Contingency-handling procedures (to be used by Subsystems). Described in DIRECTOR Note number 5.

  Console i/o procedures (to be used by Subsystems).

* Process metering and accounting procedures (to be used by Subsystems and by the system manager).

* Procedures for file or job output to local or remote devices or processors (to be used by Subsystems).

* Procedures for implementation of outward call and Subsystem system calls. DESCRIBED IN THIS NOTE.

Each procedure is reached by system call. Definitions of the relevant I, J values, and how to access them, are the subjects of a separate document.

A list of error messages is given on the last page of this Note.

Conventions

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

2. An Epage is an EMAS logical page, currently 4 kbytes. (The Epage size is said to be 4.)

3. Each user has a file index, identified by the user name, comprising six alphanumeric characters.

4. A Filename, unique in an index, may be up to 15 alphanumeric characters and may also include the character '#' or '.' (or both) except in the first position.

Since the procedures available as the EMAS 2900 DIRECTOR interface do not supply facilities for program loading and 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 collection of facilities is referred to as a Subsystem.

6. A Subsystem may choose to run "user" computations at an ACR level higher than its own, thereby effectively preventing access to some of the DIRECTOR facilities, other than via the Subsystem. For convenience in description, we assume here that all DIRECTOR procedures are invoked by a Subsystem.

---

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 EMAS 2900 System Note no 4. 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 would 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 no 4 of EMAS 2900 System Note no 4, by calling the OUT12 (defined in EMAS 2900 Supervisor Note no 4) 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.

---

ternalintegerfn DNEW INWARD CALL (integer ACCESS KEY, c
DRO, DR1, 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 DRO, 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.

J.K. Yarwood

Commands to Process Number 1, DIRECT

At System start-up process no. 1, DIRECT, performs a file system consistency check on each on-line EMAS disc pack, starts the VOLUMS and SPOOLR processes and connects the "control streams" to each FEP. Thereafter it is activated by logon messages from the interactive network, by batch job start requests from SPOOLR or by messages input from an OPER console. This note describes the last-named messages.

The command strings themselves may include space characters, although for the longest commands it is necessary to omit all spaces in order to limit the total line length. Parameters are delimited by space characters, and may not themselves contain spaces. Parameters marked with * in the descriptions below may be omitted if no ambiguity arises.

Note that <11-char filename> represents a name of not more than 11 upper case letters or digits, of which the first must be a letter.

In order to input a message to DIRECT:

press	COMMAND
type	D/<command>
press	ENTER

Commands D/PROMPT ON, PROMPT OFF may be used to initiate or terminate a "prompt" mode of input, in which a prompt "Direct:" appears on the OPER screen and an input line is despatched when ENTER is pressed.

List of Commands

Command	Page no.
ACR	3
BAD FSYS CYL TRK	3
BAD FSYS PAGE	3
BASE F	3
BROADCAST	3
CCK	3
CCK DONE	3
CIND	4
CLEAR BAD PAGES LIST	4
CLEAR FSYS	4
CLOSE	4
CLOSE DOWN	4
CLOSE USERS	4
CONNECT FE	4
DDUMP	4
DELIVER	5
DEL USER	5
DIRVSN	5
DISCONNECT FE	5
DUMPI	5
FSYS	5
INT:	5
MAIN LP	5
MSG	6
NEW USER	6
OPEN TO	6
OPEN USERS	6
PASS	6
PASS OFF	6
PRG	6
PRINT	7
PRM	7
PROMPT OFF	7
PROMPT ON	7
S	7
SEND MSG	7
SET BASEF	8
SET MSG	8
SNOS	8
START	8
STOP	8
TRANSFER	8
UNPRG	8
USERS	9
VSN	9

ACR <user> <acr> <fsys>*

Sets the ACR field in the file index for <user> on <fsys>. To be used only exceptionally. Normally, use command SETACR in MANAGR.

BAD FSYS CYL TRK <fsys> <cyl> <trk>

Makes entries in the list of "bad" (unreadable, unwriteable) pages on <fsys>. <cyl> and <trk> define 3 EMAS pages on EDS100s and 200s, and so 3 entries are created by this command. The list comprises integers holding the logical page numbers of the bad pages. The list can be printed using the command LIST BAD PAGES in ENGINR. The command BAD FSYS PAGE (q.v.) to DIRECT can be used to make a single entry of a single logical page number.

BAD FSYS PAGE <fsys> <page>

Enters (logical) page number <page> into the list of bad pages on <fsys>. See BADFSYSCYLTRK above.

BASEF <user> <fsys>*

Reports the Subsystem "basefile" identifier for <users> on <fsys>. The identifier is recorded in the user's file index. The basefile is the file which is loaded and entered by Director at process start-up. The default basefile identifier is normally null, when a file which has been "PRGed" to a fixed site, currently X'380', on the SLOAD disc is used. The basefile to be used may be specified by the DIRECT command SETBASEF (q.v.), or by a call by a user process to its Director.

BROADCAST

Broadcasts a message, set up using DIRECT command SETMSG, to each interactive process.

CCK <fsys>*

Initiates the file system consistency check for <fsys> (or for all on-line discs if <fsys> is omitted).

CCK DONE <fsys>

Paged processes cannot normally reference any <fsys> for which the file system consistency check has not been done. Exceptionally it may be necessary to allow such access, however, and giving this command enables access as though the consistency check had been done. The file system should be regarded as dangerously vulnerable unless a consistency check has been performed, and an early IPL is desirable after this command.

CIND <user> <fsys>*

Re-initialises the file index for <user> on <fsys>. To be used only exceptionally; the MANAGR commands NEW USER and DEL USER are infinitely preferable, allowing the index owners surname to be entered and an archive index to be created.

CLEAR BAD PAGES LIST <fsys>

Clears (to zero) the list of recorded "bad" (unreadable, unwriteable) pages on <fsys>.

Note: it is necessary to omit intermediate spaces in this command (except the one delimiting <fsys>) unless the "prompt" mode of DIRECT input is being used, because of the unusual length of the command line.

CLEAR FSYS <fsys>

Re-initialises the file system on disc <fsys>.

CLOSE <time>

Specifies the System close-down time (0001 to 2400) at which the service is to be automatically closed. If <time> is specified as 0 (zero), any previously specified close-down time is cancelled.

CLOSE DOWN

Causes the service to be closed (within about 30 seconds).

CLOSE USERS <time>*

Operates analogously to CLOSE <time>, except that the three System processes DIRECT, VOLUMS and SPOOLR are not stopped, allowing (for example) system maintenance work to be performed after user processes have been stopped without the need for an IPL. If <time> is omitted, the System continues to run, but new log-ons are rejected with the message "NO USER SERVICE". The service may be re-opened using command OPEN USERS (q.v.).

CONNECT FE <fe no>

May be used after a re-load of FEP software to re-establish the "log-on control stream", without an IPL of the mainframe.

DDUMP <fsys> <index no>

Used exceptionally to dump a page from the file system index area on <fsys>. Frequently command DUMPI can be more profitably used.

DELIVER

Used to set delivery information (by default equal to "Machine Room") for main log files sent to SPOOLR's LP queue as a result of DIRECT command PRINT. DIRECT prompts for an input string of delivery information, a null string causing the default to be reinstated. Delivery information remains set until a further call of DELIVERY, or until IPL.

DEL USER <user> <fsys>

Deletes a user's file index. To be used exceptionally, MANAGR command DELUSER being preferred.

DIRVSN <user> <fsys>* <vsn>*

Used either to report the Director version currently set in the user's file index (in this case one or both optional parameters are omitted), or to set the Director version (0, 1, 2 or 3) for the user to <vsn>. To be used exceptionally, for Director testing. (Supervisor command DIRVSN should also be noted in this context.)

DISCONNECT FE <fe no>

Disconnects the "log-on control stream" for the specified FEP.

DUMPI <user> <fsys>*

Used to dump out the file index for <user> on <fsys>, both in hexadecimal and partially formatted.

FSYS <user> <fsys>*

Reports the file system number on which <user> resides, or "user not known". If the optional parameter <fsys> is supplied, the search for <user> is restricted to that file system. Only discs for which the file system consistency check has been performed are searched.

INT: <user> <character>

Relays a "single-character" interrupt message to <user>, exactly as though the (Escape) INT: mechanism had been invoked from an interactive terminal. Note that if two or more processes currently exist belonging to <user> it is not possible to determine which process will receive the interrupt message.

MAIN LP

Used to direct main log output from the system to the main line printer (determined at IPL), instead of to files which are ultimately to be spooled to the LP or JOURNAL queues of the SPOOLR process. DIRECT command PRINT re-instates the spooling of main log data.

MSG <user>

Used to send a one-line text message (an operator message) to <user>. DIRECT prompts for the one-line message. Note that if two processes currently exist belonging to <user> it is not possible to determine which will receive the message. If there is no process belonging to <user> currently active, the message is stored for the next invocation of an interactive process by <user>.

NEW USER <user> <fsys> <nkb>

Creates a file index of size <nkb> for <user> on <fsys>. To be used only exceptionally, MANAGR command NEWUSER being preferred, since the latter allows for attachment of the user's surname and the creation of an archive index.

OPEN TO <usergroup>

Allows a username (6 characters) or a user group (6 characters including some '?' characters) to be appended to a list of usernames and usergroups for which log-ons are allowed even if the service is generally closed to users. Currently a maximum of 10 separate entries may be stored. Individual items cannot be removed from this list, but the whole list is destroyed by the DIRECT commands OPEN USERS and CLOSE USERS.

OPEN USERS

Allows users to (re-)commence logging on after the service has been closed (e.g. by command CLOSE USERS, q.v.).

PASS <user> <pass>

Used to specify a password by which <user> may log-on to the System (even if the service is generally closed to users). Only one <user>/<pass> combination is stored; subsequent PASS commands cause replacement of the current pair. The special password is otherwise cancelled by the DIRECT command PASS OFF.

PASS OFF

Cancels any stored special password specified earlier by DIRECT command PASS (q.v.).

PRG

Used to move System programs (supervisors, Directors, subsystems) from a file to a chosen fixed disc site. DIRECT prompts for two input lines, the first giving the <file> and <fsys>*, the second the disc <label> and disc <site>. <site> must be a X'40'-aligned numeric value.

PRINT <n>*

A value of 1 is implied if the parameter is omitted. For n>0, the current main log file is closed and copied to SPOOLR's LP queue (as well as to the JOURNAL queue).

For n>1, the subsequent n-1 main log files are additionally spooled to the LP queue (as well as to the JOURNAL queue), unless a subsequent PRINT command overrides this effect. For n=0, the current main log file is closed and spooled to the JOURNAL queue (only). Any PRINT command to DIRECT (re-)initiates spooling of main log data to disc if it was currently being printed directly on the main line printer.

Note: main log files belong to VOLUMS's file index, and commence "M#" (destined for JOURNAL) or "L#" (destined for LP). Just before any file is spooled, DIRECT renames the file (first two characters remain unchanged) so that the last four characters give the current time of day. This may help to identify files in SPOOLR's queues, for exceptional copying/printing.

PRM <file> <fsys>*

Attaches read, write and execute access permissions to <file>, specified as <user>.<11-char filename>. To be used only exceptionally.

PROMPT OFF

Cause DIRECT to remove the prompt "Direct" from the OPER screen so that the COMMAND key and "D/<command>" may subsequently be used to input commands to DIRECT.

PROMPT ON

Causes DIRECT to place the prompt "Direct:" on the OPER screen so that subsequent commands to DIRECT can be entered without the "D/" prefix and without using the COMMAND key.

S <file> <fsys>*

Reports (on the OPER screen) the logical page number of the first page of each disc section of <file> on <fsys>. <file> must be specified as <user>.<11-char filename>.

SEND MSG <user>

Provided that a message has been stored using command SET MSG (q.v.), the message is sent to <user>. Note that if two processes currently exist belonging to <user> it is not possible to determine which will receive the message. If there is no process belonging to <user> currently active, the message is stored for the next invocation of an interactive process by <user>.

SET BASEF <user> <fsys>*

Sets the Subsystem "basefile" identifier to be used by <user>, whose file index is on <fsys>. The default identifier is normally null. See DIRECT command BASEF.

SET MSG

Used to store a text message to be dispatched subsequently to one or all interactive processes by command SEND MSG or BROADCAST. DIRECT prompts (with a ":") for text input; several lines may be typed, up to a maximum of 255 characters or until a line containing only ":" is typed. Implied newline characters are replaced by spaces, in building up the stored string. When the message is complete, it is printed out in full on the OPER screen for checking before possible dispatch.

SNOS

Reports the following data on the OPER screen: the disc number from which the main supervisor was loaded, the current setting of DIRVSN (supervisor command) and the "sync1", "sync2" and "async" service number bases for local processes.

START <user> <fsys>* <dirvsn>*

Starts a process for <user>, whose file index is on <fsys> using Director version <dirvsn>. (The latter parameter is to be used only when the implications are fully understood.)

STOP

Disconnects all FEP control streams and stops the DIRECT process. (Used only exceptionally - CLOSEDOWN preferred).

TRANSFER

May be used to move a file from ownership of one user to another. DIRECT prompts for two lines of input, each containing two parameters:

<file1> <fsys1>

<file2> <fsys2>

The first determines the identity of the file to be moved; the second determines its destination. <file1> and <file2> must be specified in the form <user>.<11-char filename>.

UNPRG

Used to move System programs (supervisors, Directors, subsystems) from a specified disc site into a file (which must not already exist). Parameters are specified exactly as for DIRECT command PRG (q.v.).

USERS <usergroup> <n> or
USERS <n>*

This command is used to set limits on the numbers of interactive processes belonging to various groups of users, or to display current numbers and limits. <usergroup> is specified as a 6-character string in which '?' characters (if any) are interpreted as standing for any alphanumeric character.

USERS	<n>	sets the maximum number of interactive processes.
USERS	<usergroup> <n>	sets the maximum number of interactive processes whose owners belong to <usergroup>.
USERS		displays the current numbers in the usergroups which have been specified, with the corresponding current limits, on virtual OPER screen number 2. The corresponding display is also generated after input of the other forms of the USERS command.

VSN

Reports on the OPER screen the Director version.

J.K. Yarwood

No: 8
Date: 17/10/77

Proposed Director facilities for Testing GPC Peripherals

1. Introduction

In EMAS, all GPC peripherals are driven by interfacing to a resident routine called GPC, whose functions are:

a) allocation of devices (described in mnemonic form, e.g. CRO) to device service routines.

b) initiation of command chains expressed in RCB/LB/ALs supplied by the service routines.

c) notification to the service routines of termination, attention, program-controlled and request initiation interrupts.

Service routines may be resident or non-resident, but in either case communicate with the GPC routine by sending and receiving 32-byte messages to and from it. In addition to the 32-byte messages, work areas may be provided at device allocation time; these areas contain the RCBs, LBs and ALs and perhaps also the data areas for transfers.

The provisional specification of the GPC routine (as implemented at October 1977) is given in EMAS 2900 Supervisor Note no 5. Changes are expected in detail rather than in structure.

2. Testing of peripherals

Provision has not yet been made for general testing of peripherals, nor is design for such complete. However, the following guidelines will obtain:

a) Facilities will be provided to give one or more paged processes the level of privilege necessary to establish communication with the GPC routine.

b) Provision will be made for pages of such processes to be locked down as necessary (in particular the pages containing RCBs/LBs and ALs and the data area will clearly need to be made resident for the duration of transfers).

c) A set of procedures (part of the so-called DIRECTOR interface, and implemented as system calls) will be provided, to be used by the paged process:

i) to establish communication with the GPC routine;

ii) to allow messages to be sent to and received from the GPC routine, possibly with a modicum of validity checking of associated data;

iii) to specify the extent of necessary locking down of pages, as described above.

A provisional specification for these procedures is attached.

d) It is expected that all programs which are to run in these privileged processes will be coded in IMP (see reference), so as to afford compatible parameter passing with the procedures provided, and achieve fast compilation and good run-time diagnostics using existing, proven software.

3. Detailed specification of procedures (provisional)

(This section is subject to change.)

externalintegerfn LOCK (integer PAGE ADDRESS)
external UNLOCK (integer PAGE ADDRESS)

Causes the specified "logical page" (currently 4K bytes) of the calling process's virtual memory to remain resident (cease being resident) in main store.

Possible error results:

56 PAGE ADDRESS is not connected in the virtual memory or is invalid.

externalintegerfn GET EPAGE (integername VIRTUAL ADDRESS)
externalintegerfn RETURN EPAGE (integer VIRTUAL ADDRESS)

Allocates (de-allocates) a page of main store to (from) the process. The (local) virtual address of the page is returned in the parameter.

Possible error results:

57 (for GET EPAGE) Page not allocated to this process.

---

externalintegerfn ALLOCATE (integer MNEMONIC, integername IDEN)
externalintegerfn DEALLOCATE (integer MNEMONIC)

Makes a request on the GPC routine to allocate (de-allocate) the GPC peripheral denoted by MNEMONIC (as a multi-character IMP value, e.g. M'CRO') to (from) the calling process.

IDEN is set to an identifier to be used in execute chain requests (see below).

Possible error results:

1 Parameter invalid or device not known.

2 Device already allocated (unallocated).

externalintegerfn EXECUTE CHAIN (integer RCBA,IDEN,PAWFN,SAWFLAGS)

A 'SEND CHANNEL FLAG' is issued for the device specified by IDEN (as returned by the ALLOCATE request). RCBA gives the address of the RCB to be used. PAWFN and AWFLAGS are the PAW function and SAWFLAGS fields (right-justified in these parameters) for the controller PAW word and stream SAWO word.

Possible error results:

58 Invalid parameter or RCB/LB/ALE not accessible.

---

externalintegerfn AWAIT RESPONSE (integername IDEN, RESPO, RESP1)

The process is suspended until a response is available from the GPC. IDEN, RESPO and RESP1 are then set with the device identifier obtained from th ALLOCATE and the stream response words 0 and 1, respectively.

Possible error results:

None.

4. Conclusion

It is intended that the facilities broadly described above will enable any desired operation to be carried out "on-line" on peripheral devices not currently allocated for other purposes.

Reference

"Edinburgh IMP Language Manual", edited by R.R. McLeod (2nd Edition, 1974). Edinburgh Regional Computing Centre.

J.K. Yarwood

Operation of file system consistency check

Following IPL and the SLOAD command, a process 'DIRECT' is created, which performs the following functions:

1. file system maintenance

2. checking of log-in messages from the network and creation of other processes

3. dispatch of operator messages and broadcast messages to other processes.

'DIRECT' comprises some special routines set aside for these purposes, although it is still (currently) part of the DIRECTOR, whose code is shared by all processes.

There are four versions (0-3) of DIRECTOR on a system disc. Version 0 is the normal 'service' version, and is used by default. The DIRVSN command (e.g. DIRVSN 1) causes the DIRECTOR version specified to be used by processes started subsequently.

If the DIRVSN is 0 (the default), the DIRECT process performs the file system consistency check, starts the VOLUMS and SPOOLR processes, and connects the 'control streams' to front-end processors so that log-on messages can be received. If the DIRVSN is 1, 2 or 3 the consistency check is not done, nor are VOLUMS and SPOOLR started.

DIRVSN 1 will normally be an adequate 'back-up' for the current service version (frequently the 'previous' version); versions 2 and 3 are normally test versions.

Date and time

When the consistency check is invoked (either automatically or by command 1/CCK, which should be given only when no other process is running), the date currently set, by the DT command, is checked against the date of the previous IPL. If the current date is not within 6 days of the previous IPL, the consistency check does not proceed.

If the 'within 6 days' rule is to be overridden, either because of a previous error in 'DT' or because more than 6 days have elapsed, a specific CCK command, 1/CCK n, where n specifies the 'SLOAD' disc, must be used; in this case the date of the previous IPL will be over-written regardless. Otherwise the system should be IPLed again and a correct 'DT' given.

Consistency check error messages

A log of each consistency check is maintained on each disc, in file VOLUMS.CCKMESS. Many lines in the file commence

DT dd/mm/yy hh.mm.ss

Each username process is mentioned. Each file destroyed is mentioned. Names of files destroyed are preceded by 'FILE='.

The CCKMESS files are 'circular' text files, of size 256 Kbytes. The files may be copied to ordinary text files by the MANAGR command

CCKOUT (f/filename)

where f is the number of the disc concerned and filename is the file to contain the messages extracted. The 'f/' may be omitted, in which case all file systems are processed. 'filename' may be '.LP'.

The CCKMESS file header contains a 'READ TO' (past tense) pointer, which is updated by the CCKOUT program. If the CCKOUT program is not run before new consistency messages overrun the 'READ TO' pointer, the latter is updated with the 'START OF TEXT' pointer of the circular file header.

J.K.Yarwood

No: 10
Date: 10/2/78

Creating a new file system

On a newly-formatted disc, or where a complete file-system is to be reprimed, the following procedures should be applied. Spaces are not significant in these commands except as delimiters.

1/CLEAR FSYS n

where n is the decimal number which is the last two digits of the disc label. n may not be omitted. The response is 1/DONE. Indexes are created on the disc for VOLUMS and SPOOLR.

1/CCK DONE n

This has the effect of marking the disc table entry for the disc as being "consistency-checked". It is currently necessary only to allow the CLEAR BAD PAGES LIST command to be effected (see below). Hence it may be omitted if that command is not to be executed. The response is 1/DONE.

1/CLEAR BAD PAGESLIST n

This is necessary for a newly-formatted disc, because the count of bad pages is uninitialised. For a reprime to a disc which has not been reformatted, this may be omitted if the bad pages list is to be retained (it is not affected by CLEAR FSYS). The reponse is 1/SUCCESSFUL.

1/CCK n

A file system consistency check should now be performed on the disc.

1/NEWUSER ERCC00 n 4

should be given for as many usernames as are required. n is the FSYS number. ERCC00 is an example of the username. 4 is the number of Kbytes to be allocated to the index. Please only type 4 for the time being. You can start the reprime after 2 or 3 of these, if you start with the first users on the tape: you will be able to keep ahead! (This will not be necessary in later versions of the system - a magnetic tape will be used.)

After the reprime, an IPL and new consistency check may be a good idea, though it is not logically necessary.

Temporary

To reprime the biggest files, you may need to have done the command

```
SETMAXFILE (VOLUMS,4000)
ERCC10.DIRTESTY
```

This program will need modifying, too, to take an FSYS other than the "SLOAD" disc. You will also need file ERCC10.NEWSERVY to go with DIRTESTY.

J.K. Yarwood

EMAS 2900 DIRECTOR NOTE

No: 11
Date: 14/11/79

Director Error Messages at the Main OPER Console

This note describes messages of the form

**<username> ERROR xxx <param>

which Director sends to the main OPER console on certain error conditions, together with action to be taken (if any).

<username>	gives the owner of the process from which the message originated
xxx	are 3 hexadecimal digits identifying the error condition
<param>	If present is a decimal or hexadecimal number further qualifying the error condition

In general, these messages are of a non-urgent, non-critical nature. Special action, except where otherwise stated below, is not normally required unless the condition is persistent, when the author of this Note should be advised. In addition, notification would be appreciated of occurrences of errors marked *, which "should not occur".

Error	Condition
001*	A file is being created with the (unusual) option of being zeroed at time of creation. In the event of failure to write zeroes to the file, a re-allocation takes place. If this procedure fails 10 times, this error is given and the create operation fails.
002*	Director failed to fill stack DA entries.
003*	Director failed to find a VM gap in which to connect a segment of file indexes.
004*	Stack of ASYNC INHIBits in Director is in error.
005	Director failed to find an expected index at its correct location on disc. Action: Notify Systems staff. An early IPL and file system consistency check is indicated.
006	At DDISCONNECT, Director failed to find the correct generation of a file in the file owner's index. This can occur if a user has been deleted (DELUSER), or his file index moved to another disc, while one or more of his files were in use by another process. No adverse side effects should result.

007* Director failed to fill stack DA entries.

008 Director's file system claim-semaphore routine was called before the free-semaphore routine had been called to release the semaphore previously held.

Action: An early IPL is probably desirable to avoid possible file index corruption.

009 Director's file system release-semaphore routine was called without a preceding corresponding call of the claim-semaphore routine.

00A Accounts overflow: 1=instrs; 2=OCP; 3=pturns; 4=connect.

00B DIRECT failed to destroy a workfile belonging to a user whose process has just terminated.
PARAM = Director flag from DDESTROY.

00C DIRECT failed to find the username in its process list for a process which has just terminated.

00D DIRECT's list of active processes is full. There is currently space for 255 active processes.

00E An FCHECK file system consistency check process failed to create the consistency-check record file VOLUMS.CCKMESS.
PARAM = Director DCREATE flag.

Action: Systems staff to scrutinise VOLUMS index to discover the reason.

00F* At process start-up, the process was unable to connect a relevant disc-site (e.g. the default BASEFILE) into its virtual memory.
PARAM = flag from Director's DISC SEG CONNECT routine.

010* At process start-up, the process was unable to connect its System call table segment (for access to Director routines). PARAM = flag from Director's DISC SEG CONNECT routine.

011* At batch process start-up, the process failed to disconnect-and-destroy the file passed to it by SPOOLR giving the batch job details.
PARAM = Director's flag from DDISCONNECT.

012* At interactive process start-up, the process failed to connect its interactive terminal data streams.
PARAM = flag from Director routine CONNECT STREAM.

013* DIRECT failed to find the file index for the owner of a process just terminated.
PARAM = flag from Director's routine CONNECT INDEX.

Action: An early IPL is indicated.

014* DIRECT failed to find the file index for a valid username for whom a process was to be started.
PARAM = flag from Director's routine CONNECT INDEX.

015 At completion of the consistency check for an FSYS, the FCHECK process was unable to create a file VOLUMS.LOSTFILES in which to record filenames destroyed by the consistency check.
PARAM = flag from Director routine DCREATE.

Action: Systems staff to scrutinise VOLUMS file index to ascertain reason.

016* At completion of the consistency check for an FSYS, the FCHECK process was unable to connect the VOLUMS.LOSTFILES file in which to record filenames destroyed by the consistency check.
PARAM = flag from Director's routine DCONNECT.

Action: Systems staff to scrutinise VOLUMS file index to ascertain reason.

017* At completion of the consistency check for an FSYS, the FCHECK process was unable to connect the VOLUMS.CCKMESS file, in which a record of the consistency check is to be placed.
PARAM = flag from Director routine DCONNECT

Action: Systems staff to scrutinise VOLUMS file index to ascertain reason.

018* At completion of the consistency check for an FSYS, the FCHECK process was unable to permit the VOLUMS.CCKMESS file to DIRECT.
PARAM = flag from Director's routine DPERMISSION

Action: Systems staff to scrutinise VOLUMS index to ascertain reason.

019 DSEGMENT fails.

01A DCHSIZE new SCONNECT fails.

01B DIRECT stack extend fails.

01C REF NEW BLOCKS not zero.

01D-01F Spare.

020-035 These relate to maintenance by DIRECT of a pool of 64K-sized files for use as mainlog files by the Supervisor routine PRINTER. In each case, PARAM is a flag from the relevant Director routine. The error message is preceded by a message "LOG TROUBLE" on the main OPER console.

020 Create logfile failed.
021 Get (logfile) Disc Address failed.
022 Permit logfile to DIRECT failed.
023 Connect logfile failed.
024 Re-connect logfile failed.
025 Set file status=temporary failed.
026 Disconnect logfile failed.
027 Rename logfile failed.
028 Connect logfile failed.
029 SPOOLR logfile copy to LP failed.
02A Create and/or connect logfile copy for LP failed.
02B Disconnect logfile failed.
02C SPOOLR logfile to JOURNL failed.

02D Failed to create spare logfiles.
02E Spare.
02F Disconnect logfile failed.
030 Rename logfile failed.
031 Connect logfile failed.
032 No ready file in DIRECT's logfile pool.
033 File disc address offered by PRINTER for spooling does not belong to a file in DIRECT's logfile pool.
034* Create VOLUMS.LOGMAP file (holding DIRECT's map of logfiles and its process list) failed.
035* Connect VOLUMS.LOGMAP file failed.

036-03F Spare.

040* DIRECT fails to start VOLUMS at System start-up.
PARAM = flag from Director routine STARTP

041* DIRECT fails to start SPOOLR at System start-up.
PARAM = flag from Director routine STARTP.

042 At batch process startup, the process failed to connect the file passed to it by SPOOLR giving the batch job details.
PARAM = Director's flag from DCONNECT.

J.K. Yarwood

912 UNF ASYNC DEST is zero (more startup actions)

Format of File Index Header

```
%RECORDFORMAT HDRF(%STRING (6) OWNER, %BYTEINTEGER  %C
   FLAG, SECTSI, ALNK, USE, IMARK, %C
   IUSE, BUSE, ISESSM, SURNAME, %C
   IMAX, TMAX, BMAX, STKKB, %C
   %INTEGER LNKSTART, CELSTART, FREEBYTES, TRYING, %C
            NAMSTART, NEXNAM, LOFDAD, TOP, %C
            DWSP, BWSP, SEMA, MSGSEMA, %C
            SEMANO, IINSTRS, BINSTRS, NKBOUT, %C
            DINSTRS, IPTRNS, BPTRNS, NKBIN, %C
            IMSECS, BMSECS, CHKSUM, ASEMA, %C
   %BYTEINTEGER CODES, SIGMON, DEPTH, CONCURR, %C
            ACR, DIRVSN, MSGSPRIV, BATCHSS, %C
            BASEF, DELY, STARTF, ADRTELE, %C
            LOGFILE, SPECIALSS, SPFF1, GPHOLDR, %C
   %INTEGER USED AFDS, AFILES, AKB, USED FDS, %C
            FILES, TOTKB, ATOP, SPAO   , DATE, %C
            MAXFILE, MAXKB, DIRMON, %C
            CHERFILES, CHERKB, CONNECTT, ARESTORES, %C
            ZASL, ZFREEC, ZNCELLS, TEMPKB, %C
            FILES0, FILES1, FILES2, FILES3, %C
            BNUTS, INUTS, XLNKST, XCELST, %C
            XTOP, XNAMST, ISTOP)
```

Note: This format applies both to the on-line and the archive indexes. Each entry in the description is marked 1 if used in the on-line index and 2 if used in the archive index.

Field	Note	Description
OWNER	12	The index owner's username.
FLAG	—	Not used; previous function was to be set non-zero when an error condition (inconsistency) in the index was found. Subsequently no routine would alter or look at the index until cleared "manually".
SECTSI	1-	The (maximum) size in epages of disc sections for files described by the indexes, set up (permanently) at index creation. Currently envisaged values: 16 or 32.
ALNK	12	List-head of whole-index-permission cells in list-pool zero.
USE	—	Not used; formerly current number of processes active in this username (i.e. OWNER).
IMARK	—	Not used; intended to assist with forward/backward compatibility considerations during development.

Field	Note	Description
IUSE	1-	Current number of interactive processes active in this username.
BUSE	1-	Current number of batch processes active in this username.
ISESSM	1-	Length of current session in minutes. Zero represents the default value, currently "very large".
SURNAME	1-	Listhead of cells in list-pool zero containing the initials and surname of the index owner.
IMAX	1-	maximum number of interactive processes which may be concurrently active for OWNER. Zero represents the default, currently 1.
TMAX	1-	Maximum number of interactive and batch processes which may be concurrently active for OWNER. Zero represents the default, currently 1.
BMAX	1-	Maximum number of batch processes which may be concurrently active for OWNER. Zero represents the default, currently 1.
STKKB	1-	Number of Kbytes of the processor stack (temporary) file for invocations of the process. (This file is connected as local segment 4 during process start-up.) Zero represents the default, currently 64.
LNKSTART	12	The offset, in bytes, from the start of the index of the link bytes for pool zero of list cells. (If any of pools 1, 2 or 3 exist they commence 256, 512 and 768 bytes beyond this point respectively.)
CELSTART	12	The offset, in bytes, from the start of the index of the list cells for pool zero. (If any of pools 1, 2 or 3 exist they commence 1024, 2048 and 3072 bytes beyond this point respectively.)
FREEBYTES	12	The number of bytes between the most recent (highest) file-name (identifier) and the corresponding (and hence lowest) file-descriptor.
TRYING	—	Not used.
NAMSTART	12	The offset in bytes from the start of the index of the area for storing file-names.
NEXNAM	12	The offset in bytes from the start of the index of the position where the identifier will be put for the next-created file.
LOFDAD	12	The offset in bytes from the start of the index of the lowest (most recent) file-descriptor.
TOP	12	The size of the index in bytes.
DWSP	1-	Reserved.

Field	Note	Description
BWSP	1-	Reserved.
SEMA	1-	The semaphore word for the index (set to -1 at consistency check).
MSGSEMA	1-	The semaphore word for the owner's message-file (#MSG), set zero at consistency check.
SEMANO	1-	The semaphore number for the index, being the number of kbytes between FSYSSTART (the start of the file-system area on the disc) and the start of the index.
IINSTRS	1-	Number of thousands of machine instructions executed in interactive sessions for OWNER (cumulative over sessions).
BINSTRS	1-	Number of thousands of machine instructions executed in interactive sessions for OWNER (cumulative over sessions).
NKBOUT	1-	Number of kbytes of data "spooled out of the index" (cumulative).
DINSTRS	1-	Number of thousands of machine instructions executed in Director procedures. (Cumulative, but currently not updated).
IPTRNS	1-	Number of pageturns made on behalf of OWNER in interactive sessions (cumulative over sessions).
BPTRNS	1-	Number of pageturns made on behalf on OWNER in batch sessions (cumulative over sessions).
NKBIN	1-	Number of kbytes "spooled into the index" (cumulative).
IMSECS	1-	Number of milliseconds of OCP time attributed to OWNER in interactive sessions (cumulative over sessions).
BMSECS	1-	Number of milliseconds of OCP time attributed to OWNER in batch sessions (cumulative over sessions).
CHKSUM	-2	Checksum for the archive index.
ASEMA	1-	Semaphore word for accesses to the archive index (set -1 at consistency check).
CODES	—	Not used.
SIGMON	1-	Signal monitoring level to apply from next start-up of a process on behalf of OWNER (set zero on consistency check).
DEPTH	1-	Depth of monitor to be applied when DIRMON is non-zero (set zero at consistency check).
CONCURR	—	Not used.
ACR	1-	The ACR level at which the subsystem is to be started when a process is next created for OWNER. Zero represents the default, currently 6.

Field	Note	Description
DIRVSN	1-	The Director version (0, 1, 2 or 3) to be used at the next invocation of a process for OWNER (unless a D/START command specifies a Director version). A value of 255 represents the current default Director version (zero or as set by the DIRVSN command). When this value has been read at process start-up, it is then set to 255.
MSGSPRIV	—	Not used.
BATCHSS	1-	List-head of cells in list-pool zero to contain the name of a subsystem ("basefile") to be used for batch processes for OWNER. Currently unused.
BASEF	1-	List-head of cells in list-pool zero to contain the name of a subsystem ("basefile") to be used for interactive processes for OWNER. Currently used also for batch processes. A null filename implies use of the fixed site on the System disc. (See EMAS 2900 System Note No. 1).
DELY	1-	List-head of cells in list-pool zero to contain the default delivery information for data "spooled out of the index".
STARTF	1-	List-head of cells in list-pool zero to contain the identifier of a file of commands to be executed at process start-up. Currently unused, as the Edinburgh Subsystem keeps the identifier in the "option file".
ADRTELE	1-	List-head of cells in list-pool zero to contain the address and telephone number of the index owner.
LOGFILE	1-	List-head of cells in list-pool zero to contain a filename to be used as a Director process-monitoring file. (See EMAS 2900 Subsystem Writer's Manual).
SPECIALSS	1-	List-head of cells in list-pool zero to contain a filename to be used as a test subsystem. Reset to null filename after a process start-up using that subsystem. (Not yet implemented).
SPFF1	—	Spare list-head, currently set to ENDLIST at consistency check.
GPHOLDR	1-	List-head of cells in list-pool zero to contain a username to be used as the group-holder of OWNER'S scarce-resource units.
USED AFDS	1-	Number of file-descriptors in the archive index.
AFILES	1-	Number of files in the archive index.
AKB	1-	Number of epages in the archive index (perhaps to become the number of kbytes at a future date).
USED FDS	12	Number of file-descriptors in the on-line index.
FILES	12	Number of files in the on-line index.
TOTKB	12	Number of kbytes of data described by the index.

Field	Note	Description
ATOP	1-	Size of the archive index in bytes (currently not maintained).
SPA0	—	Spare.
DATE	1-	Compressed date and time at which a process belonging to OWNER was most recently started.
MAXFILE	1-	Maximum size (kbytes) of non-temporary files belonging to OWNER. Zero represents the default value, currently 1024 kbytes.
MAXKB	1-	Maximum total file-space (kbytes) belonging to OWNER. Zero represents the default value (currently 32768 kbytes).
DIRMON	1-	A bit-mask specifying which Director calls are to be monitored (see also the DEPTH field of this format). Set to zero at consistency check. More detail in the EMAS 2900 Subsystem Writer's Manual.
CHERFILES	1-	Number of cherished files in the on-line index (not yet implemented - the field is zero).
CHERKB	1-	Number of kbytes of cherished filespace in the on-line index (not yet implemented - the field is zero).
CONNECTT	1-	Number of seconds of terminal connect-time for interactive processes belonging to OWNER (cumulative over sessions).
ARESTORES	1-	Number of files restored from archive (cumulative over sessions).
ZASL	12	Free-list pointers for list-pools 0, 1, 2 and 3.
ZFREEC	12	Numbers of free cells in list pools 0, 1, 2 and 3.
ZNCELLS	12	Numbers of cells in list-pools 0, 1, 2 and 3.
FILES 0-3	-2	Number of types 0-3 files described by the archive index. Type 0 is archive files, type 1 is back-up files. Types 2 and 3 are unassigned.
INUTS	1-	Number of scarce-resource units assigned to the index.
BNUTS	—	Reserved for possible use as batch-resource.
XLNKST XCELST XTOP XNAMST	12	Copies of LNKSTART, CELSTART, TOP and NAMSTART (copied at consistency check), present as a prelude to moving the semaphore fields (SEMA, MSGSEMA, ASEMA) nearer to the front of the index.
ISTOP	—	Dummy entry marking the end of the record format.

J.K. Yarwood

FEP

EMAS 2900 FEP NOTE

No: 1 (revised)
Date: 2/5/79

The EMAS 2900 Front End Processor

Contents

1. Introduction

2. Software Components

   2.1 AM1 Handler
   - 2.1.1 "Internal" transactions with the 2900
   - 2.1.2 Data transactions with the 2900
   - 2.1.3 Interface to the high-level protocol tasks
   - 2.1.4 AM1 handler fault messages

3. Operation of FEP

   3.1 Loading procedure

   3.2 Powering off procedure

   3.3 Dumping the system

   3.4 Processing a Front End dump

   3.5 Responding to crashes

1. Introduction

This note describes the system software running in the EMAS 2900 Front End Processor, which is a DEC PDP11/34. It also describes operating procedures.

Section 2 summarises each software component in turn; subsection 2.1 then described the AM1 handler in greater detail. It is intended that further subsections will be added in due course, each likewise dealing with a single component in detail.

Section 3 is taken largely from the EMAS 2900 Operator's Manual. It describes the operational aspects of the FEP.

Figure 1.1 below indicates how the software components of the FEP are interlinked, and, for the ERCC 2970 installation, how the FEP is connected to the EMAS 2900 mainframe on the one hand and the Regional Computing Organisation (RCO) Network on the other. The protocol hierarchy for interactive and non-interactive operation is shown diagrammatically in Figures 1.2 and 1.3, and for completeness the formats of the various protocol packets are also given (Figure 1.4).

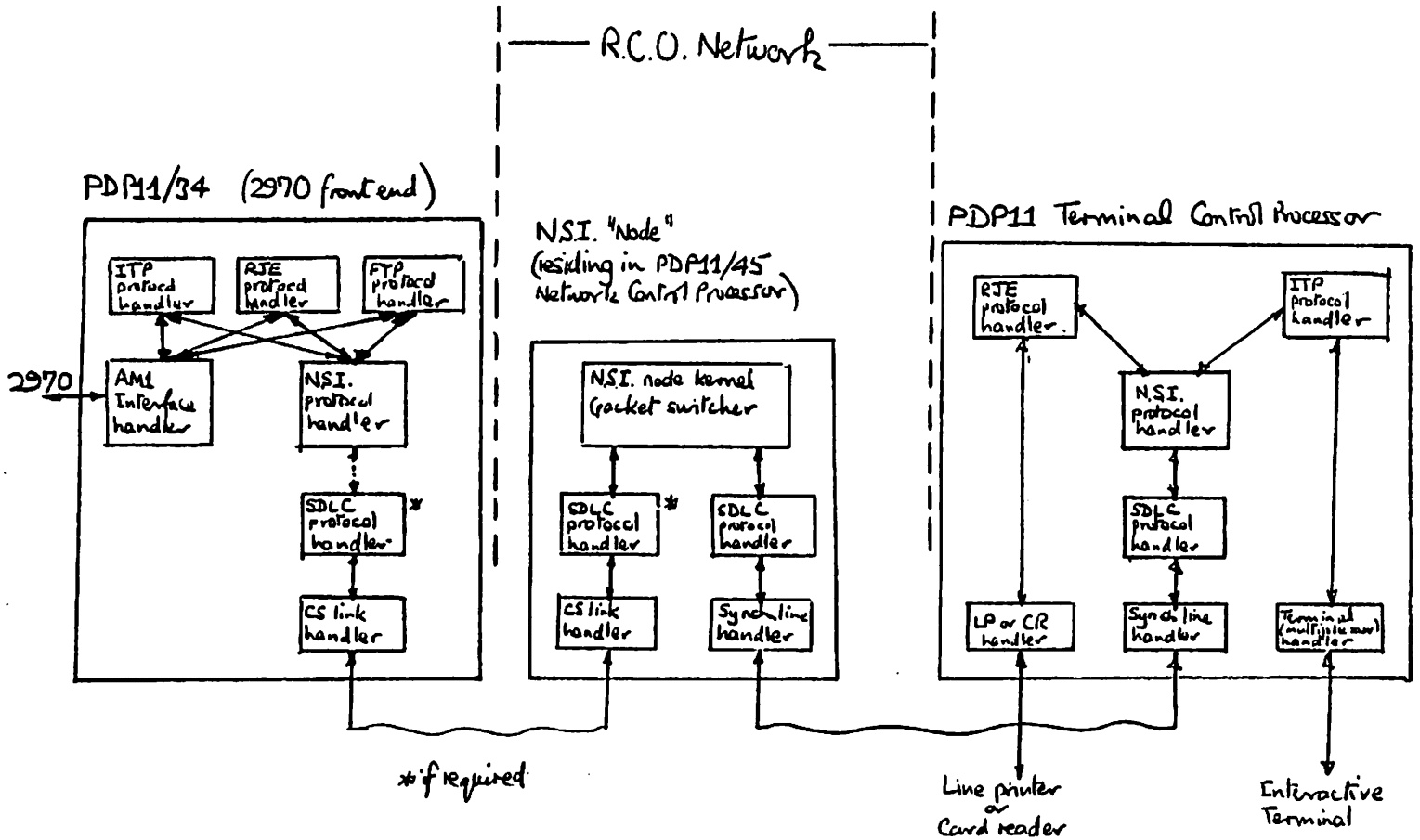


Figure 1.1: Hardware and Software Configuration for ERCC 2970

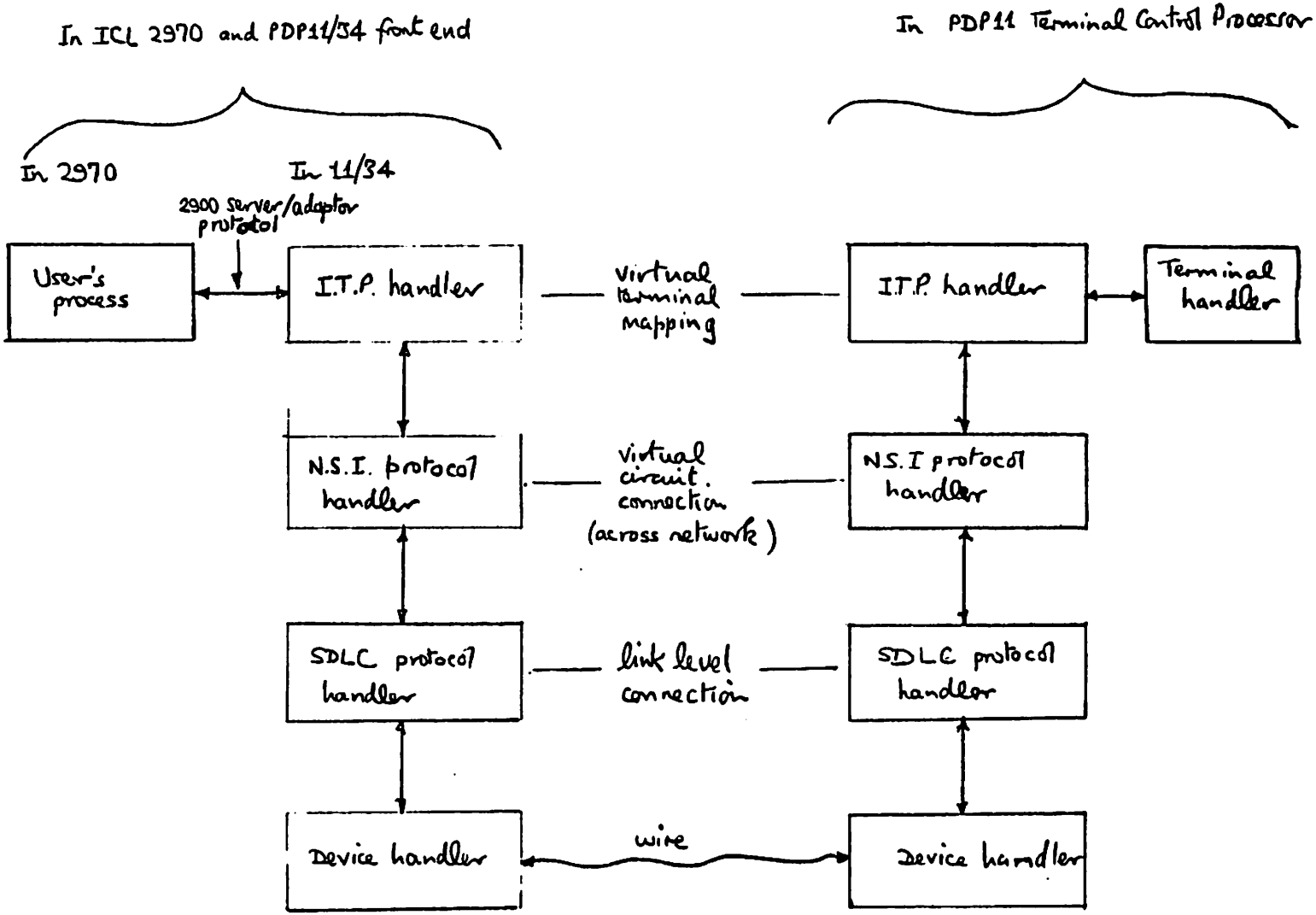
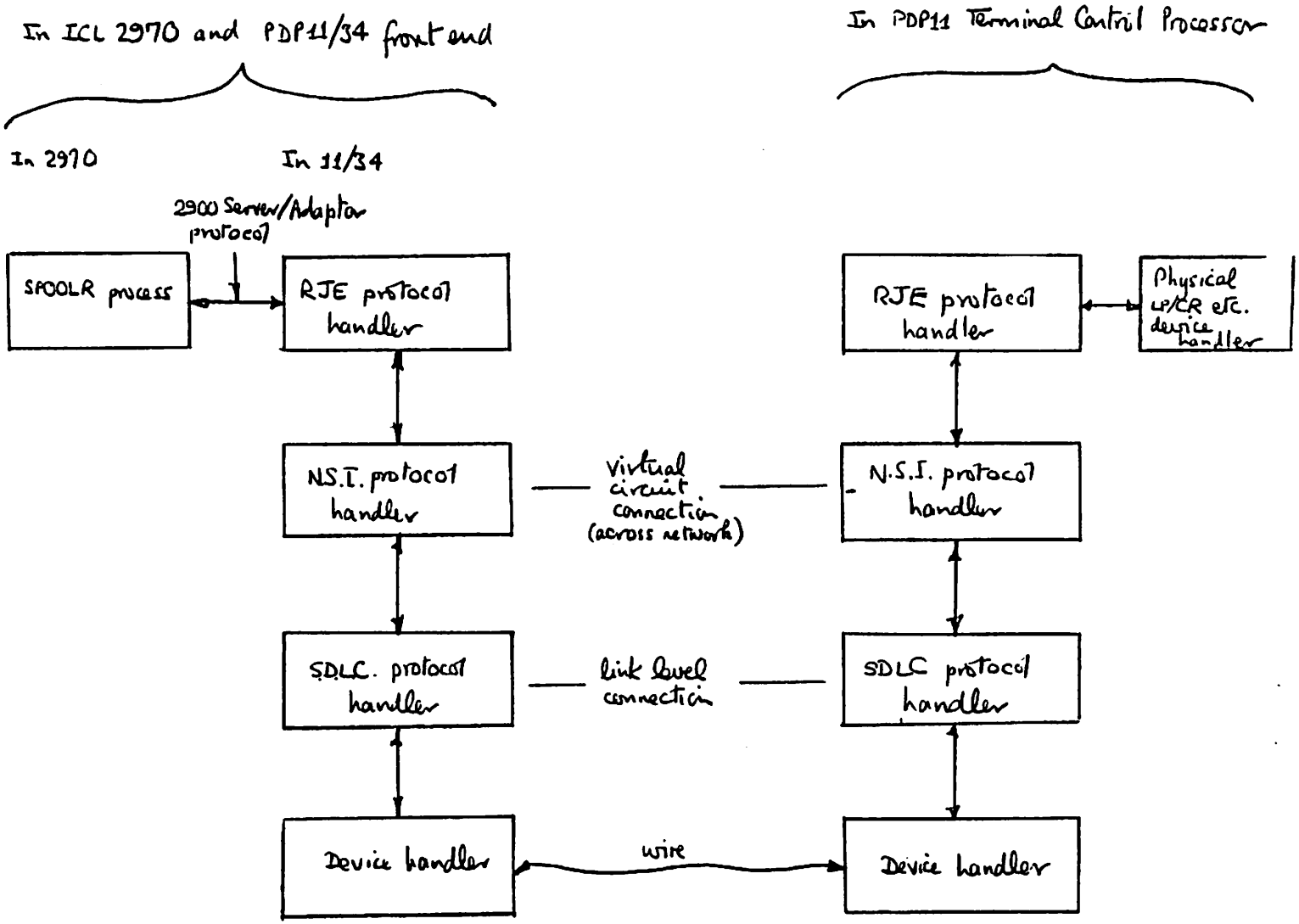


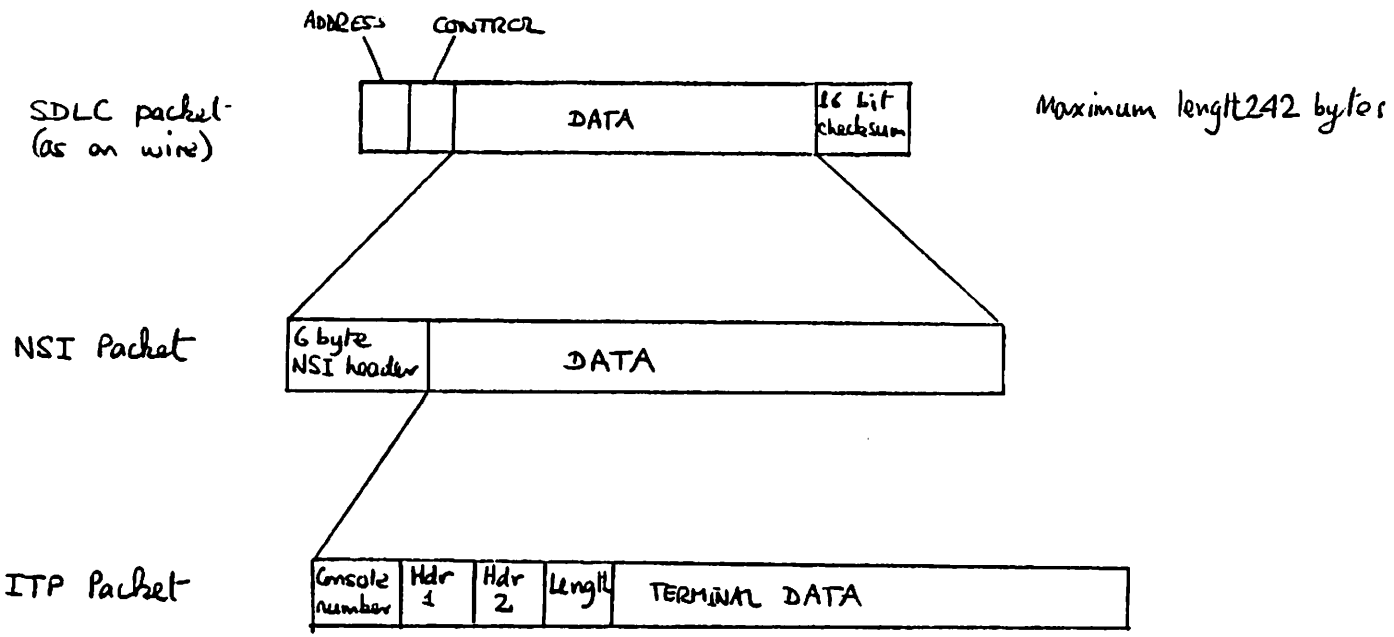
Figure 1.2: RCO Network Protocol Hierarchy - Interactive



Notes: N.S.I. — Node Standard Interface.
The structure for a File Transfer Protocol (FTP) is essentially the same, although it will normally be used for mainframe/mainframe communication.

Data flow — ⟷

Figure 1.3: RCO Network Protocol Hierarchy - Non-interactive



Notes: I.T.P. – Interactive Terminal Protocol
N.S.I. – Node Standard Interface

Figure 1.4: RCO Network - Packaging of Interactive Data

2. Software Components

The software consists of a number of co-operating programs, each running as a task under the operating system DEIMOS. (DEIMOS is a general purpose operating system designed for use in PDP11s with memory management.)

The left hand part of Figure 1.1 above shows the connections between the programs which drive the FEP. The arrows indicate the interfaces between the tasks. In addition all the tasks interface to the buffer manager and to the common TT output handler.

It should be obvious that nearly all the above tasks are critical; e.g. if the NSI handler crashes, then everything will stop. The only exception is the RJE handler, the failure of which will only bring down the RJE part of the service.

Each of the programs in the 2900 FEP is now summarised in turn.

1) AM1 Handler [Task name: AM1H]

The AM1 handler handles the "protocol" necessary on the AM1 interface. For example, the 2970 will inform the handler that it is prepared to make a transfer on a stream 'n'. The handler determines what use the stream is put to, and instructs the appropriate protocol handler. The ITP handler, for example, will then read (or write) characters from the 2900, enclose the ITP packet headers round it and pass the "message" to the NSI interface module. This module adds on the NSI packet header, and passes the "message" to the CS link handler for transmission into the Network.

2) Interactive Terminal Protocol (ITP) Handler [Task name: ITPS]

This program handles the protocol-dependent features of the ITP For example, after receiving user output from the 2900, it will add the protocol header to the output and, obeying the protocol flow control rules, will pass the output to the Network Standard Interface (NSI) protocol handler, for transmission into the RCO network.

In addition, the handler controls the virtual connections along which the data to and from each Terminal Control Processor (TCP) flows. Each TCP sets up one connection to the FEP along which all terminals connected to that FEP communicate.

3) RJE Protocol Handler [Task name: RJES]

This handler controls all remote file transfers to and from the EMAS 2900 mainframe.

Data formatting and flow control on each virtual connection is performed by this module. Messages from operator consoles, i.e. logons and queue enquiries, are passed through to the mainframe with the protocol headers removed. Messages to operator consoles are processed into the correct form and passed to the NSI handler.

4) Network Standard Interface (NSI) Protocol Handler [Task name: NSIH]

This handler processes the protocol-dependent features of the RCO NSI, passing the information contained within to the high-level protocol modules (ITPS and RJES), in a near network-protocol-independent manner. For example, when a high-level protocol accepts a data packet from the handler, it forms the network-dependent "SEND BLOCK RESPONSE" packet and transmits this through the network. In the case of a different protocol, e.g. X25 level 3, the handler would construct a "window change" or 'RR' packet from the same response from the higher level.

5) Computer Science Asynchronous Link Handler [Task name: CSLK]

The handler receives blocks from the NSI handler and transmits them, one character at a time, down the link. It blocks up the character record from the link and passes them in a block to the NSI handler. In addition, in contains watchdog timers to establish when the link goes down.

6) File Transfer Protocol (FTP) Handler

This module is still to be written; its function will be to transmit and receive files from the network in the EPSS Study Group 3 FTP protocol.

7) Buffer Manager [Task name: BUFF]

This program controls all the data buffers used in the FEP system. All other tasks send requests for buffers to the buffer manager and return buffers to it after they have been used.

8) HDLC Protocol Handler [Task name: PROT]

On systems which communicate to the network via a synchrous line an HDLC protocol handler is used instead of the CS link handler. This module performs the same functions as CSLK, but ensures an error-free path by implementing HDLC at the link-to-link level.

For details of how these tasks communicate with each other, and other features of the FEP system, consult the DEIMOS User Manual, by B.A.C. Gilmore (ERCC).

2.1 The AM1 Handler

The AM1 handler controls the PDP11 hardware interface with the 2900 mainframe. In the case of the 2900 P series the interface is via the GPC, with 2900 S series via the DCU. The handler also controls the low-level functions of the software "protocol" that is used to communicate between a 2900 and a PDP11.

The handler has three distinct functions:

1) To handle "internal" transactions with the 2900

2) To handle data transactions with the 2900

3) To interface to the higher-level protocol tasks

These are described in turn below (subsections 2.1.1 - 2.1.3).

2.1.1 "Internal" transactions with the 2900

There are a number of control signals identified by bit 0 set in the receive status register sent out by the 2900 which are completely handled by this task. The response to each is defined either by the 2900 interface hardware or software.

The functions, and their responses, are:

a) Send Property Codes

This is sent to the PDP11 on every IPL of the 2900, and is used by the 2900 hardware to establish how many front ends (if any) are currently available.

The response is 4 data bytes, followed by a normal primary status:

14,0,0,0, 16 + control bit.

The '14' defines a front end to EMAS 2900.

b) Identify

Identify is produced by the GPC microprogram (in a P series 2900) to determine which of the possible mechanisms on the interface has set an "attention bit" in primary status. This is not actually used in the FEP case, as there is only one mechanism, but nevertheless the identify must be replied to.

This response is two data bytes and a normal primary status:

0,X'80', 16+control bit.

c) Sense

This command is normally generated by EMAS 2900 software to obtain the secondary and tertiary status bytes after certain conditions, e.g. send property codes. Currently no "real" information is actually transferred to the mainframe. Response: 2 data bytes and primary status:

0,0, 16+control bit.

d) Limit

Limit is sent by the 2900 when it has read the number of bytes specified in its current command. The action taken depends on the state of the handler. If it is handling one of the above control signals, e.g. identify only expects two data characters (so that a limit is seen before the primary status can be transmitted), then a primary status is sent and earlier, still pending, actions are ignored.

In the second case, the interface will be in a "user pending or writing to the 2900" state (see below). The action in this case is still to send a primary status.

There are other possible control signals, e.g. initialise, autoload, which are not used by EMAS 2900. If any is actually received, the response is a primary status with "unsuccessful" set.

2.1.2 Data transactions with the 2900

All data transactions to or from a 2900 consists of two parts. First, a four-byte part describing the logical stream which the transfer is to to be on (this also determines the direction), and the maximum length of that particular transfer. This four-byte "header" always goes from the 2900 to the FEP. The second part of the transaction is the actual data; this can be transferred in either direction.

From the point of view of the handler, the actual transaction is as follows:

a) A write control command signal, followed by four data bytes - two for the stream number and two for the length of transfer, followed by a limit command - to cate the end of that part.

b) The handler must now send a primary status to the 2900, to signal the completion of the command.

c) The 2900 will now send either a READ command or a WRITE command, depending on the direction of the transfer. A number of data bytes is thus transferred in the relevant direction and the command again terminates by a primary status.

There are two distinct types of data transfer: a control stream transfer, and a transfer on any other streams. The distinction is only made for the convienience of handling control stream transfers (see below) which are done by the AM1 handler itself.

A control stream transfer (stream numbers -1 and -2) is done in units of 24 bytes. In the case of an outward transfer, the 2900 sends up to 8 units simultaneously. These are read by the handler, the stream number to which they refer is determined, and the entire message is then passed to the higher level protocol which "owns" that particular stream. In the case of inward transfers, the higher level protocol tasks send messages containing 24 bytes to the handler which queues them and, when permitted, sends up to 8 of them into the 2900 in any one transfer.

Control of transfers

If the link is idle, the 2900 may at any time start up a transfer to the FEP. If the FEP wishes to initiate a transfer to the 2900, it must send an ATTENTION to the 2900, which should then respond with a request to do an inward control transfer. If a transfer is in progress, the FEP may indicate its desire for a transfer by setting the ATTENTION bit in the next (and subsequent) primary statuses that it sends to the 2900.

Note: All transfers from the FEP to the 2900 are initiated with a control transfer on a stream that requests the 2900 to initiate an actual data transfer on the particular stream. Several of these may be outstanding at any time, and it is the prerogative of the 2900 to decide which it honours first.

Stream transfer

When a transfer is initiated on a stream other than the two control streams, the handler passes physical control of the link over to the higher level protocol task and lets it do the actual transfer. When it is finished, or when the other task senses a command on the link (e.g. a LIMIT), it passes control back to the handler, which performs the primary status.

2.1.3 Interface to the high-level protocol tasks

There are six messages from the handler to a higher task (parameters in brackets):

0	Pass interface address (physical address of interface)	
1	Do input (stream, max trf)	do data transfer from 2900
2	Do output (stream, max trf)	do data transfer to 2900
3	Message (address of message)	control message from 2900
4	Mainframe up	state of mainframe
5	Mainframe down	state of mainframe

There are five messages from high level protocol tasks to the AM1 handler (parameters in brackets):

0	Send high level control message (address of message)	to be sent on inward control stream
1	Send low level control message (address of message)	to be sent on inward control stream
2	Here I am (stream no)	claim of stream no
3	Return control (action)	pass control of link back to the handler
4	Stop	tells the handler to close down

There are a number of actions that can be performed on "return control". The values and actions are:

0	Send normal primary status	
1	Send normal primary status + short block	significant information transferred, e.g. EOF
2	Send normal primary status + long block	transaction not satisfied - 2900 should put another one on
3	Send unsuccesful primary status	
4	Send primary status + condition	
5	Same as 0	

In addition, if the top bit of the flag is set, then the handler should accept the test characters sent from the 2900; otherwise it should not.

2.1.4 AM1 handler fault messages

The AM1 handler produces error messages (on the FEP console) of the following form:

AM1: FAULT x n1,n2

where: x is a character indicating the class of fault
n1,n2 specify a subclass and/or give extra information

Note: except where stated otherwise, "input", "output", "receiver" and "transmitter" are with respect to the PDP 11/34.

Classes

x= T - Timeout (character level) - a character from the 2900 was expected (i.e. in the middle of a sequence) but did not arrive within 20 ms (approx).

n1= state of receiver side of AM1
n2= state of transmitter side of AM1

x= I - Input fault

n1= 1 - after initial WRITE CONTROL (w.r.t. 2970), four characters should be received before a limit; n2 gives the number of characters actually received.

2 - a transfer request has been made by the 2970, on a stream that has not been CLAIMED by a higher level task; n2 is the stream number.

3 - a READ (w.r.t. 2970) has been issued on a stream >0 but a number of non-control characters have also been received; n2 is the number of characters in error.

4 - same as 3 except a WRITE (w.r.t. 2970) has been issued.

5 - an unexpected control character has been received from the 2970; n2 is the control character.

6 - non-control characters have been received before a control character; n2 is the number of characters.

x= S - A CONTROL message (high or low level) has been issued to a stream that has not been CLAIMED by a higher level task.

n1= number of offending stream
n2= offset into the entire block received from the 2970

x= O - Unexpected OUTPUT interrupt. An interrupt has been generated on the transmitter side of the AM1 - this should not have occurred.

n1= 0
n2= output state of AM1

x= E - Transmitter state fault. The transmitter side of the AM1 has got into an error state.

n1= output state of AM1
n2= previous output state of AM1

x= R - Return control fault. A RETURN CONTROL has been received from a higher level task at a time when the AM1 handler has not released control.

n1= input state of AM1 (w.r.t. 11/34)
n2= output state of AM1 (w.r.t. 11/34)

x= C - Clock timeout (or major timeout). A request for transfer on the control stream, or an output interrupt, has been outstanding for 15 seconds. This will generate a message to the higher level tasks informing them that the 2970 has gone down.

n1= 2 - transfer outstanding
n2= 1 - interrupt outstanding

3. Operating the EMAS 2900 Front End

The following description relates specifically to the operation of the FEP on the ERCC 2970.

The EMAS 2900 Front End Processor, a DEC PDP 11/34, is connected to both GPCs on the 2970 mainframe by a locally produced NRPI link. Figure 3.1 illustrates this connection.

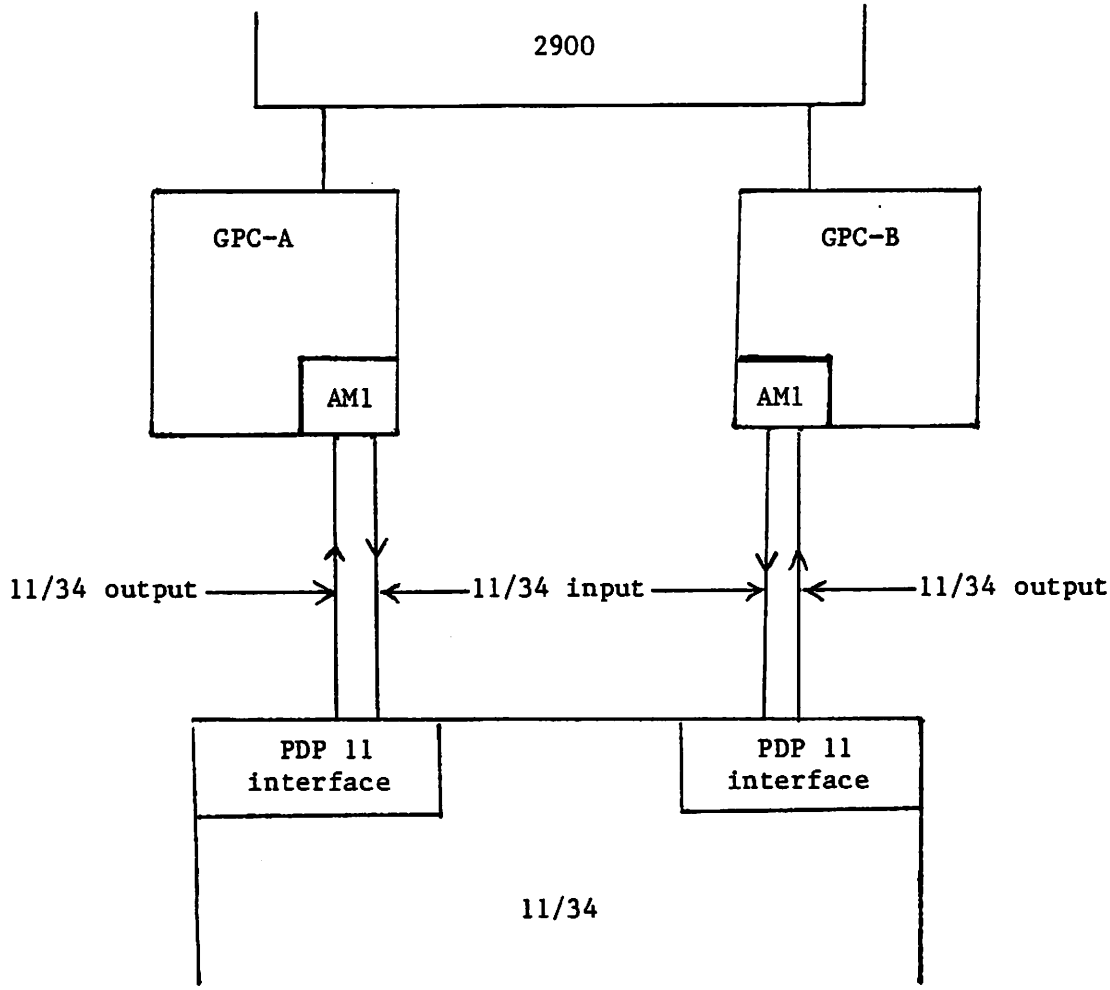


Fig. 3.1: Connection of FEP to Mainframe

As explained above, the FEP is linked to the 2970 via the AM1 interfaces. It uses only one of these - the other is for backup. The FEP is also linked to the PDP 11/45 Network Control Processor via a 1 Megabit Computer Science asynchronous link. All interactive and RJE traffic to the 2970 uses this link.

3.1 Loading Procedure

The procedure is as follows:

* If the RUN light is on, hold the CTRL button down and press HALT, then release the CTRL button.
* Boot the machine: hold the CTRL button down and press BOOT, then release the CTRL button.
* The bootstrap should now identify itself with something like the following (the actual numbers do not matter):

  010000 160200 170000 000002

To load the system from disc drive 0:

* Ensure the LOAD/RUN switch is at RUN.
* Type DK (cr).
* After a pause the system identifier "DEIMOS VSN 6" appears, and after a further pause of a few seconds the system prompt "%" appears.

The programs that drive the link etc. must now be loaded, by typing:

LOGON 15 (cr)

When the prompt reappears, type either

GPCA (cr) or GPCB (cr)

depending on which connection to the mainframe is currently in use.

The Front End is now ready for use.

Notes:

* In this section: (esc) = ESCAPE, (cr) = RETURN, _ = space.
* Ensure that both GPCs are switched on.
* "ATTACHED OK" indicates that the NCP (11/45) has recognised the 11/34.
* Each TCP, as it "connects" to the 11/34, produces a message of the form:

  TERMINAL_x_CONNECTED

* After a reload of the NCP (11/45), connection of the 11/34 must be re-established by typing

  (esc) INT_R_CSLK (cr)

3.2 Powering Off Procedure

1) The Front End should be closed down by typing on its console:

   (esc) INT_D_GATE (cr)

   The reply

   GATE: REMOVED OK

   should be received.

2) The Load/Run switch on the disc should be switched to LOAD.

3) The machine should be halted by holding down Ctrl and pressing Halt.

4) Finally, power the machine off.

3.3 Dumping the System

A dump should be taken if the 11/34 stops; that is, if the run light goes out. The sequence for taking a Front End dump is:

* Holding CTRL down, press HLT.
* Press CLR.
* Press 1 then 4 then 0.
* Press LAD.
* Holding CTRL down, press START; the disc lights on unit 0 should blink and the run light will go on then off again.
* Fill in the front end dump log and process the dump (See 3.4).

3.4 Processing a Front End Dump

A front-end dump always occupies the same site on the RK05 system disc and it is important to process this dump immediately before reloading the service system, as any future dump would overwrite the one already on the dump site. Three other sites are reserved on the system disc to hold processed dumps and these should be used cyclicly and their use recorded in the Front End dump log. The three sites are referred to as DMP001, DMP002 and DMP003.

The procedure is as follows:

* Load the Deimos operating system, as in 3.1, until the first system prompt appears.

* Log on to user number 16 rather than user 15:

```
%LOGON_16
```

* Process the dump using the program DPAL and the following replies to its prompts:

```
% DPAL_/DMPnnn       where nnn specifies the next dump site

DPAL: T
DPAL: M
DPAL: Q_47
DPAL: Q_51
DPAL: Q_52
DPAL: S
```

* Continue with the normal load as described in 3.1 by logging on to user number 15.

3.5 Responding to crashes

The 2970 FEP will normally run without the need for operator intervention. However, under certain circumstances action is necessary.

1. GPC TIMEOUT + FEO DOWN on the 2970 Oper

   a) If this message is accompanied by

      AM1: Fault C 2, 0

      EMAS 2900 DOWN

   on the FEP console, then a dump of the FEP is <u>not</u> necessary, but

      (esc)INT_F EAM5

   should be typed on the FEP console, to obtain monitoring information. Once it finished, a reload of the 2970 will be necessary, but not a reload of the FEP.

   b) If a message of the form:

      xxxx: BAD SEGMENT - followed by two numbers or
      xxxx: ADDRESS ERROR or
      BUFF: NO SMALL BUFFERS *** or
      BUFF: NO BIG BUFFERS ***

   has been printed on the FEP then a <u>dump</u> (see 3.3) <u>is</u> necessary.

   N.B. The last two messages above can appear without the system crashing; a dump should only be taken if the FEP is not responding to users.

2. If no users can log on to the 2970 and the 11/45 front end to the 4/75 link is up

   In this case, the 2970 FEP should be dumped and then reloaded.

B.A.C. Gilmore
J.M. Murison

2970 FEP - COMPILING NEW VERSIONS (1)

All the Fep modules are on file system 15, so a LOGON 15 should be done before editing or compiling a module.

To compile a module, the command is:-

IMPS source,.TT/object

If the 1st pass of the compiler succeeds, - n statements compiled, the compiler proceeds to the 2nd and 3rd pass, after the end of the 3rd pass, - code = n bytes etc., the LINKING stage will prompt for data:-

NAME:

STACK:

STREAMS:

The date typed is different for each module and is given below in the form:- GATE,300,1 - each bit, i.e. 'Gate' is typed in on a line by itself.

Modules (name of source and object file is given)

A) AM1 Handler

There are two versions,

1) EAM5/EAM5Y - this is the unmodified interface handler
2) EAM5X/EAM5XY - this is the modified interface handler

in both cases the linking parameters are:- EAM5,300,1

B) ITP protocol handler

ITPSR/ITPSRY,parameters ITPS,500,1

C) RJE protocol handler

RJESR3/RJESRY parameters RJES,300,1

D) NSI Protocol handler (Gate)

EGATE/EGATEY parameters GATE,300,1

E) C.S. Links handler

CSLK3S/CSLK3Y parameters CSLK,200,Ø

F)/

F) Buffer Manager

BUFFR/EBUFFY parameters BUFF,200,Ø

G) Common Output Handler

COMMS/COMMY parameter COMM,200,Ø

Note: For security, the original object file should be transferred to file system Ø before you compile the new file

T RJESRY/RJESRY(Ø)

and if the new object fails, the old object file can be recovered by:

T RJESRY(Ø)/RJESRY

EMAS 2900 2970-FEP LISTING SOURCE MODULES

The best way of obtaining a listing of a source module after it has been compiled is to produce an 'ALIST' (see USER manual) of the program and list it on the Computer Science line printer.

1) ALIST ITPSR,ITPSRY/.LP - to alist the ITP handler.

2) Type (esc)T LR6/LP on the Computer Science ISYS system.

EMAS 2900 2970-FEP LISTING A PROCESSED DUMP

After a FEP is processed (see operating instructions) it can be listed on either the CS line printer or on a network line printer, this should be done before the Dump slot is overwritten.

A) Via CS Line Printer

1) LOGON 16
2) T DMP001/.LP - do list DMP001
3) on the CS ISYS machine, type (esc)T LR6/LP

These commands may be typed while the normal system is running, but 'escape' must be typed to produce the prompt '*' before each command.

B) Via the network

This must be done while the normal system is running.

1) (esc)RJELPY DMP001(16) - to list DMP001
2) the program will then prompt:-
   a) Node: - type ∅ and return
   b) TERM: - type 23 and return to list on the slow devices LP
   c) FACILITY: - type 4 and return

When LP23 becomes free, it will print the dump, with delivery to B.GILMORE.

EMAS 2900 FEP

AM1 HANDLER MONITORING OUTPUT.

The AM1 handler (EAM5) outputs the list 255 transactions whenever it thinks that the 2900 has gone down (version of 10APR.79). It can also be requested to output the information by typing:-

(esc)INT F EAMS

The format of the monitoring is a single character and a number, the meanings of each character are:-

C n - a control character 'n' has been rec'd from the 2900

I n - a data character 'n' " " "

D n - a data character 'n' has been sent to the 2900

D -n - a control character 'n' has been sent to the 2900
note: this always comes as a pair with a D n,
e.g. D 16)
D -16) → primary status X'20' has been transmitted

S n - 'n' is the stream number that the 2900 is going to do a transfer on, it is purely 'internal' monitoring, rather than a character transmitted or rec'd.

M n - a 'Poff' rec'd by the task, the values of n determine what
n=243 - input interrupt
=242 - output interrupt
=1&2 - control message from a higher task to go to 2900
=3 - a higher task returning control to EAM5

X n - a 'limit' has been rec'd and the current O/P sequence is terminated.
n=current state (This is a normal event on an identity).

U n - EAMS has re-asserted 'accept chr' after the 2900 has accepted the primary station, n is meaningless.

W n - In a timeout sequence, n is the timeout count.

Example of monitoring output from the AM1 link handler showing

1) an ATTN sent to the 2900 and its setting up of an inward control transfer.
2) at the end of 1), it then asychronously starts up a transfer to stream 13.
3) the handler accepts this and passes control to the relevant user program.

```
D   -32)
D    32)  attn
M     0
C    10   identify
D     0
D   128
M     0
C    12   limit
X     4
M     0
D   -16)
D    16)  terminate
M     0
C     5   write control
I   255
I   254
I     0
I   192
C    12   limit
S    -2   monitoring
D   -16)
D    16)  terminate
M     0   INT
C     2   READ (wrt 2970)
D     0
D    11
D     0   data (control stream=
D    10   24 byte)
D     0
D     1
D     0
D   186
D   255

D   255
D   255
D   255
D     0
D     0
D     8
D     0
D     0
D     0
D   150
D     0
D   129
D   129
D   129
D   129
D   -16
D    16   terminates NB is only one byte
M     0   inst (now M 243)
C     5   write control
I     0
I    13
I    11   input data
I    70
C    12
S    13   monitor of stream
D   -16)
D    16)  term
C     3   write (Mus 2970→11/34)
M    42)  return control from    (N.B. from
2     3)  10/4/79 this comes as M 3)
D   -16
D    16   terminates
U-13680   accepts last character sent
```

2960 FEP - NOTE ON 2970 DOCUMENTATION

1. Where '.LP' is specified - i.e. in 'LISTING SOURCE MODULES'. .TT will have to be used.

2. The source module of 'GATE' is called KGATE

Edinburgh Regional Computing Centre

Memorandum

To R. Eager

Please pass to Alan afterwards.

From George Howat

Date 14-10-81 Extension 2614

Ref.

Ref.

TCP Setmode Command Changes

It is acknowledged that the current implementation of setmodes (Local and Remote) contains deficiencies and irritating features. Therefore, in the next TCP Software release, I propose to alter existing and add "new" setmode functions. This will involve changes in all server and/or front-end software and documentation for mainframes which implement the ITP setmode control mechanism, (I believe that only the 2976 and INFO will remain unaffected), The modifications are not too drastic. Alterations to current practice are described below. For completeness, the appendix summarises the full RCO set.

User Terminal Interface

Three pairs of "new-format" commands are proposed:

i)	V, - V	:	enter, exit video delete mode
ii)	G, - G	:	" " graph mode
iii)	F, - F	:	" " flow control mode (XON/XOFF output)

Note:

i) Is an unambiguous set video-delete mode, removing the present "toggle"

ii) Currently, exit from graph mode also involves entering upper-case mode and exit from binary input mode. The three functions graph, binary, upper-case are now to be decoupled.

iii) Gives the user access to XON/XOFF output mode locally at the terminal i.e. can be used on any system which does not implement ITP setmode - this has been requested by Glasgow (who cannot - will not send setmode commands) and users of micros emulating terminals.

All other user setmode commands are unaltered but their action is modified e.g. "U" imposes upper case only; binary input and graph mode flags are unaltered.

Mainframe Commands

The above changes have impact on the mainframe software: both bytes in video and graph commands (15 and 11 respectively) are now significant (see Appendix). The EMAS user interface will also have to be changed e.g. GRAPH = ON/OFF; G, - G etc.

The new command to be introduced at the request of the EMAS team, command value 28, is a "RESET DEFAULTS", see appendix.

Comments and/or suggestions to me please.

Appendix: Summary of Setmode Commands (changes marked with asterisk)

User	Remote Command	Value(s)	
-	0	-	Not used
-	1	0/1	Echo off/on
H 2-255 (decimal)	2	5-255	Set page size and enter page mode
W 15-160 (decimal)	3	15-160	Fold line after n characters
-	4	-	Return current parameters (core map)
R 0-255	5	0-255	Set data forwarding character
I 0-255	6	0-255	Set escape character
D 0-255	7	0-255	Set character delete character
C 0-255	8	0-255	Set buffer delete character
-	9	0/1	Binary input mode off/on
T 1-160... (decimal or *)	10	1-160...	Set tab position (seven values only)
* G,-G	11	0/1	Graph mode off/on
L	12	-	Select upper & lower case
* U	13	-	Select upper case only
P 0-10 (decimal)	14	0-10	Insert n pad characters after carriage return
* V,-V	15	0/1	Video-delete mode off/on
-	16	0-255	Append character to server prompt
* F,-F	17	0/1	XON/XOFF output mode off/on
-	18	0-255	Enter bulk input mode with stop char in 2nd byte
-	19-22	-	Used by Kent.
-	23	0/1	SCREED mode off/on

24-27	0-255	4 byte bit mask
28	-	Reset default modes

Notes:

'-' as a 'value' means not significant. Under the 'User' column, means cannot be set locally by the user.

Default modes and parameters:

- Echo on
- Page mode off
- Fold line after 80 characters (currently 72)
- Data forwarding character = CR
- Escape character = ESC
- Character delete = DEL
- Buffer delete = DC3
- Binary mode off
- Tab values: 1, 6, 9, 12, 15, 18, 40, 80
- Graph mode off
- Upper case only
- Zero pad characters
- Video-delete off
- No append to prompts
- XON/XOFF mode off
- Bulk input mode off
- SCREED mode off

GRAPHIC OUTPUT FILE FORMAT

Each file comprises 3 header records followed by the actual plotting data all in fixed 80 byte records.

Header Blocks

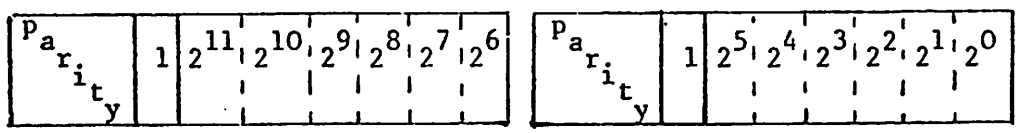
1. 80 EBCDIC 'X' characters
2. 80 ISO characters - of which the first 40 are typed on the PDP-8 teletype preceded and succeeded by CRLF

BYTES	CONTENTS
1-3	spaces
4-11	jobname, eg 'ERCC05'
12-18	spaces
19-26	data, eg 10/06/74
27-30	spaces
31-38	time, eg '14:53:54'
39-40	spaces
41	device code, 'P' = plotter 'B' = binary & selectric
42	plotter code, '0' when byte 41 = 'B', otherwise '1' = ERCC Calcomp 936 '2' = Social Science 30" '3' = " " 11" '4' = Calcomp 564 0.1mm '5' = Calcomp 564 0.005in.
43-44	2 digit paper length requirement if >10ft or 3m, otherwise set at '00'
45	delivery information indicator X'FF' ≡ no information 'A' - 't', ie 65-94 (N-64) characters of user's address information
46-49	spaces
50-75	spaces if byte 45 = X'FF', otherwise (N-64) delivery information characters
76-80	spaces

3. 80 EBCDIC 'Y' characters

Data Blocks

BYTES	CONTENTS
1-2	data block count modulo 4096 - starts at 1 - in the form

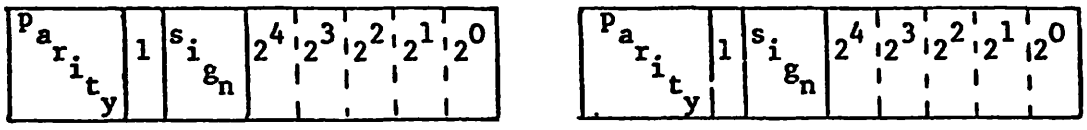


3-4	
5-6	
.	
.	data
.	
79-80	

The data may be 1, 2 or 4 bytes in length, dependent upon the next vector type.

a. Short Vector

2 bytes - which can cope with vectors <32 increments long, in the form



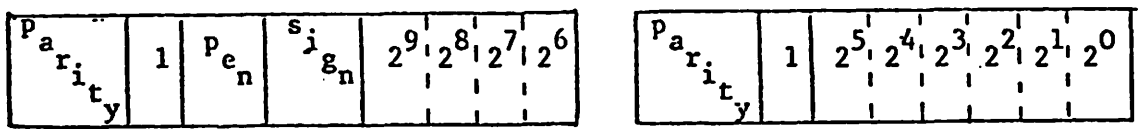
ΔX (2's complement) ΔY (2's complement)

i. possibly preceded by 1 byte (value 29_{10}) to specify CHANGE PEN STATUS (up/down) if required prior to this move. and

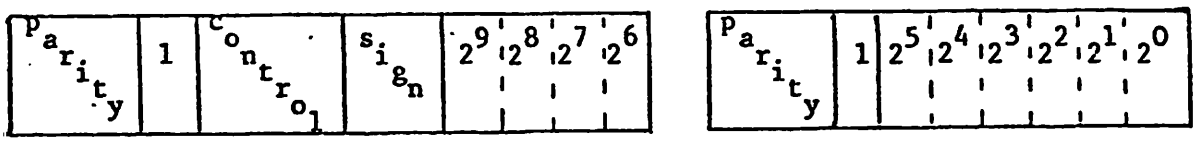
ii. also possibly preceded by 1 byte as a signal (value 31_{10}) to specify CHANGE TO SHORT VECTORS.

b. Long Vector

4 bytes - which can cope with vectors <1ϕ23 increments long in the form



ΔX (2's complement)



ΔY (2's complement)

i. possibly preceded by 1 byte as a signal (value 30_{10}) to specify CHANGE TO LONG VECTORS.

In ΔX the P_{e_n} bit indicator is

ϕ raise pen before moving)
1 lower pen before moving) if $C_{o_{n_{t_{r_{o_l}}}}}$ below $\equiv \phi$

and in ΔY the control bit indicator is

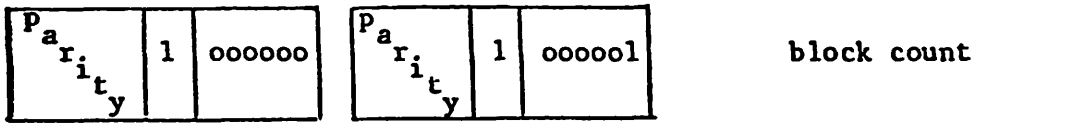
ϕ ≡ these 4 bytes are a LONG VECTOR

1 ≡ these 4 bytes are a CONTROL SEQUENCE when the ΔX value means

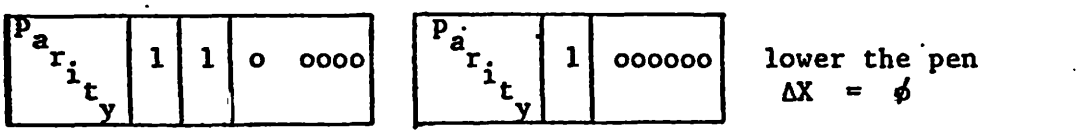
ϕ ≡ new sub-file (plotter origin change)
1 ≡ change pen to black biro
2 ≡ " " to blue "
3 ≡ " " to green "
4 ≡ " " to red "
5 ≡ " " to ϕ.1mm liquid ink
6 ≡ " " to ϕ.2mm " "
7 ≡ " " to ϕ.3mm " "
8 ≡ " " to ϕ.4mm " "
9 ≡ " " to ϕ.5mm " "
10 ≡ " " to ϕ.6mm " "
11 ≡ " " to ϕ.8mm " "

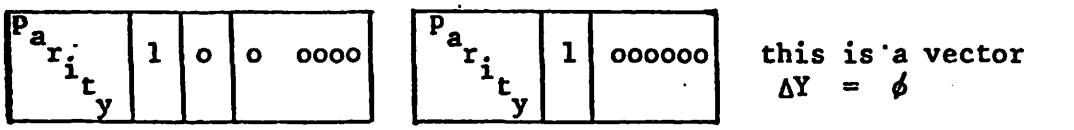
NOTES

1. Each plotter file has in the first 7 bytes of DATA BLOCK 1

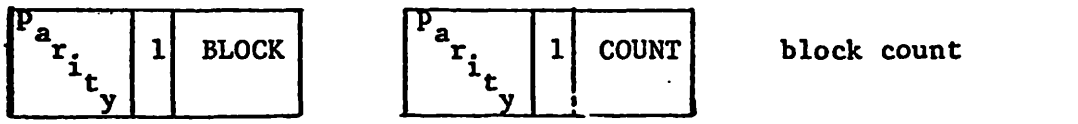




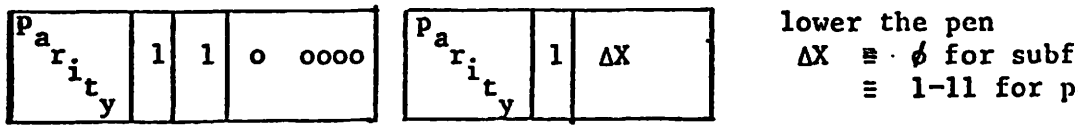


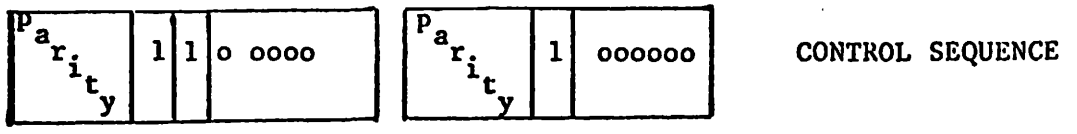


2. Each sub-file or pen colour change request starts on a fresh record with the first 7 bytes containing









3. Only complete 1, 2 or 4 byte groups will appear on any record - there will be no splitting between records

4. On a subfile, or pen change request, or when a 1, 2 or 4 byte grouping will not fit, the current record is padded out with the prevailing VECTOR TYPE INDICATOR (31 for SHORT VECTORS, 3Ø for LONG)

5. SHORT VECTORS will be used whenever possible. 2 SHORT VECTORS will be used in preference to 1 long one when the move is <63 increments long.

M.D. BROWN
16 January 1976

REGIONAL COMPUTING ORGANISATION
PLOTTER FILE STANDARD

C O N T E N T S

		Page
1.0	Introduction	2
2.0	Data Structure	2
3.0	Bytes	3
4.0	Commands	3
	4.1 Vector Commands	3
	4.2 Directive Commands	5
5.0	Records	9
6.0	Steps	9
	6.1 Vector Steps	10
	6.2 Directive Steps	10
7.0	Files	10
8.0	Programming Standards	10
	8.1 Generating Software	10
	8.2 Interpreting Software	11
Appendix		13

Regional Computing Organisation
Plotter File Standard

1.0 INTRODUCTION

The Technical Advisory Panel of the RCO set up a working party, initially to advise on the desirability of a standard for plot files, and subsequently to produce a standard. This report gives the proposed standard. This standard is necessarily subject to continuing review as experience is gained and new facilities implemented.

The design goals were:

1. To allow packages running on any computer within the region to be plotted on any plotter.

2. To allow files intended for a particular plotter to be diverted to another plotter with a minimum of distortion.

3. To ensure that the format of the file is sufficiently flexible to allow plotters with advanced features to be incorporated later.

4. To have a consistent and easily implementable format.

A file standard is not sufficient to ensure portability; there must also be operational and programming standards. Some attempt has been made to include these, but further work will be necessary. As an example the standard does not specify what should happen if a plot exceeds the size of the plotter.

2.0 DATA STRUCTURE

There is a five level hierarchy of information within the plotter file. In increasing order of complexity, the five levels are:

1. Bytes

2. Commands

3. Records

4. Steps

5. The file

The plotter file consists of steps, which consists of records, which consist of commands, which consist of bytes.

3.0 BYTES

The basic unit of representation is the 7-bit byte. If 8-bit bytes are sent, the most significant bit will be ignored. The information bits within a byte are numbered from right to left, with bit 0 being the rightmost (least significant) bit, and bit 6 the leftmost (most significant) bit.

In byte strings representing characters the ISO standard character set is used.

4.0 COMMANDS

All commands consist of a control byte normally followed by data bytes. No command is split across a record boundary (see below). There are two classes of commands - vector commands and directive commands.

4.1 Vector Commands

These are commands to draw visible and invisible vectors. Both absolute and relative vectors are allowed. A vector command consists of a control byte followed by from 0 to 6 data bytes.

4.1.1 Control Byte -

The control byte has the following form:

0	A	V	X_1	X_0	Y_1	Y_0
6	5	4	3	2	1	0

Bit 6 is always zero.

A-field: This is a one bit field. The vector is absolute if this bit is one, and relative if it is zero.

V-field: This is a one bit field. The vector is visible (pen down) if this bit is one, and not visible (pen up) if it is zero.

X-field: This is a two bit field containing a count of the data bytes that follow for the X co-ordinate of the vector. Values from 0 to 3 are allowed.

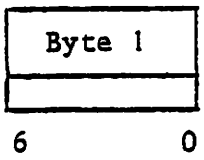
Y-field: This is a two bit field containing a count of the data bytes that follow for the Y co-ordinate of the vector. Values from 0 to 3 are allowed.

Note: Data bytes are stored following the control byte as X co-ordinate data bytes, if any, followed by Y co-ordinate data bytes, if any.

4.1.2 Data Bytes -

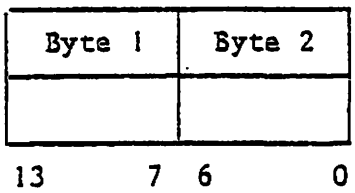
X (or Y)=0: There are no X (or Y) data bytes. For action see note below.

X (or Y)=1:



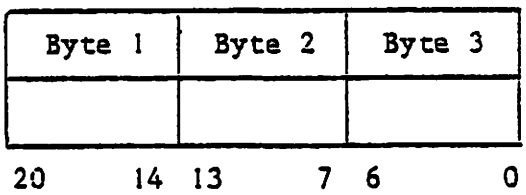
X (or Y) co-ordinates stored as sign in bit 6 and modulus in bits 5 to 0.

X (or Y)=2



X (or Y) co-ordinates stored as sign in bit 13 and modulus in bits 12 to 7 and 6 to 0.

X (or Y)=3



X (or Y) co-ordinates stored as sign in bit 20 and modulus in bits 19 to 14, 13 to 7 and 6 to 0. (see § 1.1)

4.1.3 Notes - the following byte sequences have the effects indicated.

1. 00(8) Raise the pen without changing position.
2. 40(8) Raise the pen and move to origin.
3. 20(8) Lower the pen without changing position.
4. 60(8) Lower the pen and move to origin.

4.2 Directive Commands

A directive command consists of a control byte, a count byte (containing the number of data bytes), and data bytes if appropriate. The only exception is the null command (4.2.2.1), which has no count or data byte.

4.2.1 The Control Byte

The general form of the control byte is:

1	n5	n4	n3	n2	n1	n0

The control byte for a directive always has bit 6 = 1, and has a 6-bit type code N.

4.2.2 Commands -

4.2.2.1 N=0, The Null Directive -

The null directive is used for padding out records. It is not followed by a count or data bytes.

1. control byte: 100(8)

4.2.2.2 N=1, Start of File Directive -

1. control byte: 101(8)
2. count: (12 + size of account record) <76
3. data bytes:

   byte 1: override byte

   = 1 plot the file even if the required hardware facilities do not exist

   = 0 abandon plotting a file if any hardware facilities do not exist

   bytes 2,3,4: maximum x extent of plot in plotter steps.

bytes 5,6,7: maximum y extent of plot in plotter steps

bytes 8,9: number of steps/unit

byte 10: plotter units:

= 1 centimetres

= 2 inches

byte 11: number of simultaneous pens or linetypes required.

byte 12: hardware character set:

= 0 none required

> 0 required type of hardware character set (currently undefined)

bytes 13- n: (n < 76) user name/account left justified.

The start of file directive must be contained in a directive step (see § 6.2).

4.2.2.3 N=2, Start of Step Directive -

1. control byte: 102(8)
2. count byte: 8
3. data byte 1: step number (modulo 128)
4. data byte 2: overlay byte:

   = 0 non overlaid step

   > 0 overlaid step
5. data bytes 3-5: maximum x value of frame in plotter steps
6. data bytes 6-8: maximum y value of frame in plotter steps

Note: This must be the first directive on the first record of the new step.

4.2.2.4 N=3, Pen/Linetype Selection Directive -

This directive selects the pen or line type required for subsequent vector drawing. Pen/Linetypes are numbered consecutively from zero to (n-1), where n is specified in byte 11 of the start of file directive (§ 4.2.2.1).

1. control byte: 103(8)

2. count: 1

3. data byte: 0-7 selects pen 0 to pen 7

8-127 not yet defined

4.2.2.5 N=4, Pen/Line-Definition Change Directive -

This directive carries pen/line information for those devices that can utilise it. ~~It also serves to suspend plotting when operator intervention is required to change pens, etc.~~ It is a directive to modify the table of pen option entries. The previous directive specifies a physical pen change.

1. control byte: 104(8)

2. count: 4

3. data byte 1: pen type/speed

= Ø - ballpoint pen

= 1 - ink pen

= 2 - felt tip pen

4. data byte 2: colour/tone

= Ø - black

= 1 - red

= 2 - blue

= 3 - green

5. data byte 3: width

= Ø - 0.1 mm

= 1 - 0.2 mm

= 2 - 0.3 mm etc.

6. data byte 4: pen number. Pen position or line type referenced above.

The definition of this directive is subject to changes in the details of the data bytes. The directive is used to redefine pen/line characteristics. The actual implementation of the required change will be organised in various different manners depending on the actual hardware involved.

4.2.2.6 N = 5, Operator Message not Requiring Intervention -

This types a message on the operators console and continues plotting.

1. control byte: 105(8)
2. count: 1-72
3. data bytes up to 72 ISO characters to be typed

Note: No format effectors should be included in any message.

4.2.2.7 N=6, Operator Message Requiring Intervention -

Similar to 4.2.2.6. but plotting is suspended until the operator signals it to continue.

1. control byte: 106(8)
2. count: 1-72
3. data bytes: up to 72 ISO characters to be typed

Note: No format effectors should be included in any message.

4.2.2.8 N=7, Hardware Character String Directive -

This is a string of characters to be plotted, and is intended for plotters with hardware character sets. Plotters without hardware characters may or may not simulate them.

1. control byte: 107(8)
2. count: 1-72
3. data bytes: up to 72 ISO characters to be plotted

Note: See Section 8.1.9

4.2.2.9 N=8, Hardware Character String Parameters Directive -

The format of this directive has not yet been defined. It is intended for setting up parameters such as size and orientation on plotters with hardware character sets.

1. control byte: 108(8)

2. count: not defined

3. data bytes: not defined

4.2.2.10 9<=n<=31 Reserved for Future Use -

These directives have not yet been defined, and are reserved for enhancements to the regional standard.

4.2.2.11 32<=n<=62 Reserved for Local Use -

These directives are reserved for local use, and it is permissible for interpreting software to ignore them.

4.2.2.12 N=63, End of Plot Directive -

1. control byte: 177(8)

2. count: 0

This directive must be the final directive in the terminating directive step (§ 6.2).

5.0 RECORDS

A record consists of 80 bytes. The first byte is the step number, the second the record number within the step modulo 128. Commands may not be split across a record boundary, any unused bytes at the end of a record being padded with null directives. Records are numbered from 0 and are in strictly ascending sequence. A special catagory of record with a step number of zero and record number of Ø constitutes a directive step (see § 6.2).

6.0 STEPS

A step consists of a number of plotter records, and is a logical subdivision of the plotter file. For example, a plot file consisting of several graphs might be divided into steps such that each graph is a separate step. In some implementations a step may be a restart point.

A step may be overlaid on a previous step. In such cases the first step should occupy the largest frame.

6.1 Vector Steps

A vector step commences with a step directive (§ 4.2.2.3). Each record within a step contains the step number in the record header. Thereafter the record may contain both vector and directive commands. A vector step may be overlaid on a previous step. In such cases the first step should occupy the largest frame. Steps are numbered from 1 in strictly ascending order.

6.2 Directive Steps

Directive Steps are numbered Ø and do not contain a start of step directive. They are essentially records with step number Ø and record number Ø and contain only directive commands (see for instance §4.2.2.2 and § 4.2.2.12).

7.0 FILES

The plot file may contain header and trailer information (such as produced by HASP). The plot information starts with a record of 80 bytes with values Ø to 79. This is used for framing the file into records in stream oriented input systems. The second record consists only of (4.2.2.2) a directive step holding a start of file directive command.

8.0 PROGRAMMING STANDARDS

Programming standards are to allow the recording of standard practices as these are found desirable through experience of various implementations of the Plotter File Standard. No list should be assumed to be exhaustive and implementors should check the current status.

8.1 Generating Software

To date the following standards to be observed by software generating plotter files are established.

8.1.1 Many interpreters will only accept vectors up to 15 significant bits (+ sign bit). Hence any larger vectors should be generated as a succession of such 15 bit vectors. An absolute vector greater than 15 bits can be generated as an absolute zero (invisible) plus a succession of invisible vectors to the required point.

8.1.2 The initial default setup should be assumed by any generating software. See Appendix 1.

8.1.3 Whenever steps in a plot are to be overlaid the first step will be assumed to occupy the largest frame.

8.1.4 Each step should close by positioning forward for the next step.

8.1.5 Each file should contain a message not requiring operator intervention displaying user name, job name and delivery information on the operators console.

8.1.6 Generating software should cause the same information to be plotted on the output device, if appropriate.

8.1.7 Operator messages should not contain carriage control information. It is the responsibility of the interpreter to format such messages.

8.1.8 A negative absolute vector should not be generated.

8.1.9 Every hardware dependent operation involving a pen movement should be followed by an absolute pen positioning command.

8.1.10 Vectors should be represented by the smallest number of vector data bytes possible.

8.2 Interpreting Software

Interpreting software will be as diverse as the devices available for plotting. The following paragraphs are intended to clarify the action of interpreters in areas where implementations may be radically different, in addition to defining standards for interpreters to follow.

8.2.1 Pentype Selection

The number of simultaneous pens/line types is limited by the number requested in the start of file directive. For example, if three pens are requested only pen/line types 0,1 and 2 may be used. Within this restriction any pen type may freely be selected (4.2.2.4) by the generating software. If it is necessary to use a greater number of pen types, one of the permitted pens must be redefined by the change pen directive (4.2.2.5). When implementing generating software note that most interpreters will only handle satisfactorily a small number of simultaneous pen types (see default setup in Appendix 1).

8.2.2 Interpreting Pen Type Selection

Normally the interpreter will accept as many pen types as are available on the hardware. However it is feasible for the interpreter to accept a greater number, when the interpreting software must organise the changing of pens as different pen types are selected. In this case, the pen select command may cause plotting to stop while the operator is instructed to change the pen. This is not feasible if pen changes are frequent!

If a greater number of pens are requested than the interpreter can handle, and it is decided to plot the file anyway, it is invisaged that all 'extra' pen types would be treated as, say, pen Ø.

8.2.3 Pen/Linetype Definition Change

This command redefines one of the available pens or defines pens additional to that assumed in the default setup. It is the business of interpreting software to organise the mapping of different pen/linetype definitions onto the physical pens or linetypes available on a particular device. Any operator intervention required is organised by the interpreting software.

APPENDIX 1

DEFAULT DEVICE SETUP

Software generating files to this standard and satellite software interpreting these files to a graphic device, should both assume that the following pen/line type initial conditions apply at the start of each file, (ref. Pen/Line Change Directive).

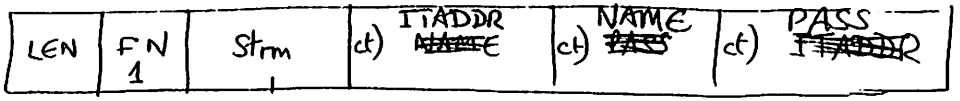
	Output Device	
Pen/Line Type	Graph Plotter	Tektronix VDU
Ø	Black biro	Solid line
1	Red biro	Dotted line
2	Blue biro	Chained line
3	Green biro	Short dashed line
4	-	Long dashed line

Commands creating these conditions will not appear in the file; it is the task of satellite software to reset these conditions at the start of each file if necessary.

Messages DIRECT ↔ FEP

In to DIRECT

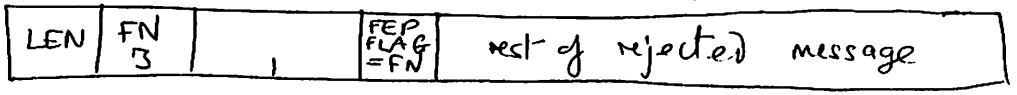
① Logon



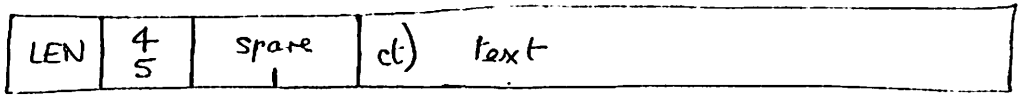
② Mode data



③ rejected message

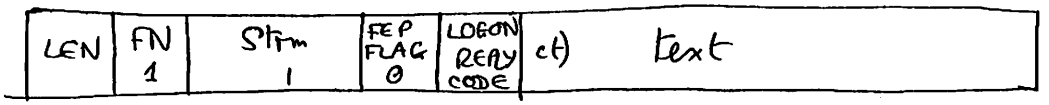


④ ⑤ } monitor msg to { Mainlog / Oper

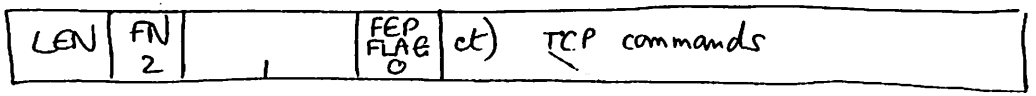


Out from DIRECT

① Logon reply



② Setmode



TCP Action codes on a SETMODE call

Byte 1	
Ø or 16	End of setmode sequence
1	If Byte 2 = Ø then echoing off, else on
2	Set page length to byte 2 lines
3	Set line length to byte 2 chars
4	Bulk 'getmode'. Returned as "ITP control + setmode" transfer
5	Set CR char to byte 2
6	ESC
7	DEL
8	CAN
9	If byte 2 = Ø then "binary input mode" off, else on
10	Set tabstops, seven values in bytes 2 - 8
11	Graphical o/p mode on/~~off~~
12	Lower case input allowed
13	Turn off graphmode, binary input mode, lower mode
14	Set number of pads to byte 2 (=N nulls after CR)
15	~~Toggle~~ video mode on/off (same as 23)
16	Above
17	Flow control (1 = on, 0 = off)
~~17~~,18,19	Not used at Kent. NOP's
20	Preset input buffer. Byte 2 = count. Bytes 3 on = chars to be loaded into i/p buffer. Use for em 'open'
21	Byte 2 = Ø then tabs are expanded into spaces on i/p, else tabs are sent as themselves to host.
22	Set all user options in one go - bulk setmode
23	Reliable video mode. Byte 2 = Ø for off, else on

N.B. setmodes can be strung end to end within one transfer if so desired. No terminator is required, since the TCP effectively appends a Ø byte to your transfer.

Bulk setmode/getmode variables

N.B. preceded by a 22 for setmode

Byte 1	Flags 1.	B:Ø not used B:1 Echo flat 1 = don't echo B:2 Graph mode 1 = on B:3 Lower mode 1 = on B:4 Binary i/p mode 1 = on B:5:7 Not used B:0 Flow control (1 = on)	'Not used' bits undefined on getmode. Masked out on setmode
Byte 2	Flags 2.	B:Ø:2 Not used B:3 Video mode 1 = On B:4 Not used B:5 Hardware tab i/p mode 1 = On B:6 Raw mode 1 = on B:7 Not used	
Byte 3	Pad count		
Byte 4	Spec says physical line number - in fact it is undefined		
Byte 5	Linelength		
Byte 6	Pagelength		
Byte 7	Undefined (in fact it is always 1)		
Bytes 8 - 14	Tab vector 7 bytes		
Byte 15	CR char		
Byte 16	ESC char		
Byte 17	DEL char		
Byte 18	CAN char		
Bytes 19 - 34	Raw mode mask (128 bits) Least sig bit byte 19 for NUL (ØØØ) Most sig bit byte 34 for DEL (177) bit = 1 means treat as line break char. If B:6 of byte 2 = Ø, this mask may be omitted on a bulk setmode.		

Use of Raw Mode

Raw mode is selected by setting B:6 of byte 2 in a bulk setmode (action code 22). Bytes 19 - 34 of the setmode define a 128 bit mask, one bit per ascii character. Setting a bit to Ø indicates that the character corresponding to that bit position is to be buffered on input and not immediately sent to the host. Raw echo will be performed by the TCP, but no translation will be performed - e.g. TABs will not be expanded into spaces. When the user types a character whose corresponding bit in the mask is set to 1, all buffered input together with that character will be sent to the host. The "line-break" character will not be echoed by the TCP.

The maximum input line which can be buffered is 120 characters. Should the user type 120 consecutive non-line-break characters, the TCP will send the buffer to the host.

To avoid problems in some ITP implementations, CR is mapped into CR+128.

The INT: character is recognised at all times. This offers the user some protection against programs which "die" in raw mode. The character can be changed via byte 16 of a bulk setmode if programs are really desperate.

Should a program rely on multiple line-break characters, it should be aware of ITP goahead restrictions. If the TCP runs out of goaheads it will reject user input and eventually force a "console failure" disconnect.

All output from the host is passed to the user with no translation. When the user inputs a line-break character, further echoing of non-line-break input is deferred until the host has responded with output. This guarantees the state of the user's VDU for programs such as screen editors.

A. L. Ibbetson
April 1981

File Transfer

Operations Command set

The Command

S/OPENFT (S/CLOSEFT)

given without parameters is the 'main switch' of the File transfer mechanism. This opens (closes) the file transfer activities of EMAS2900 in general and, in the case of open, allows transactions with all other stations which have been allowed activity.

An individual station is allowed activity by the command

S/OPENFT address

or all stations by

S/OPENFT .ALL

Such commands can be given at any time but only take effect if (or when) the 'main switch' (OPENFT with parameters, as described above) has been given.

Example: S/CLOSEFT

Sequence:

S/OPENFT	KENT		
S/OPENFT	DEC10	----	no activity permitted
S/OPENFT		----	KENT and DEC10 can go ahead
S/CLOSEFT	DEC10	----	only KENT can now go ahead
S/OPENFT	.ALL	----	all stations can go ahead
S/CLOSEFT		----	activity is not permitted
S/OPENFT		----	all stations can go ahead

S/MON 3 — switches on FTP monitor.

S/FTP ADDR

S/FTP S/R user user remote pass file.

The command

S/FTLINES

allows operational control of file transfer activity in terms of the number of transactions in progress at any one time.

A transaction either started life initiated by the home station (a 'master' transaction) or was initiated by an external station (ie, a 'slave' transaction). There must generally be available some 'slave' capacity, which in the default situation consists of half the available transaction lines.

Hence

S/FTLINES 6

will allow up to 6 transactions to proceed but 3 of these transactions will wait to be initiated by external stations ('slave' transactions) and the remaining 3 will serve the home file transfer queue ('master' transactions)

and

S/FTLINES 6 2

will allow up to 6 transactions to proceed of which 2 will wait to be initiated by external stations.

Note that this specification of lines to be initiated by external stations does not imply a limit to the number of lines which can be used by external stations: any unused line can be used by an external station. However in general the home station has priority whenever contention occurs.

The maximum number of transactions in progress (master or slave) for any particular external station can be set by

S/FTLINES address n (eg, S/FTLINES KENT 2)

Note that one extra slavewill be permitted over this value to prevent home station hogging which can occasionally occur.

FTLIMIT

The command

S/FTLIMIT

allows operational control of file transfer activity in terms of the files size of transactions sanctioned.

S/FTLIMIT n

limits all transactions to n k

S/FTLIMIT address m

units all transactions to the specified station to mk (subject to m n)

Hardware

EMAS 2900 HARDWARE NOTE

No: 1
Date: 17/2/78

Information on 2900 Image Store

(Received verbally by J.K. Yarwood from Alan Darling and Bill Kell, 4th May, 1977; updated June 1977; updated with the help of Pat Ryan January 1978.)

The purpose of the discussion was to establish differences between the 2980 and the 2976, 2970 & 2960 with a view to trying EMAS 2900 on the former. The following table summarises the model-dependent addresses and uses for the four machines:

	2960.Bits	SML16 2970.Bits	2976.Bits	SML12 2980.Bits
OCP Registers				
LST Base	00006001	00006001	402A	402A
LST Limit	6000	6000	402C	402C
PST Base	6003	6003	4029	4029
PST Limit	6002	6002	402B	402B
Handkeys	6006	6006	4205	4205
Hooter (set bit to commence hoot; clear bit to terminate hoot)	6008 31 (Optional)	6008 31	4013 19	4013 19
System Interrupt Register (Bit set to indicate interrupting Port)	600A 28-31 for Ports 0 to 3	600A 28-31 for Ports 0 to 3	4014 22-27 for Ports 0 to 5	4014 22-27 for Ports 0 to 5
Real time clock (X)	600C	600C	44pt4000	44pt4000
Real time clock (Y)	600D	600D	44pt4100	44pt4100
Real time clock (Z)	600E	600E	44pt4200 (see footnote)	44pt4200 (see footnote)

	2960.Bits	SML16 2970.Bits	2976.Bits	SML12 2980.Bits
OCP Control Registers				
Control (inhibit when set)	6011	6011	4013	4013
Clear slaves - write only (reads 0)	0			
Inhibit CPRs (Hamming reports)	24	24		
Inhibit main store		25		
Inhibit retry		26	13	13
Inhibit pipeline		27	12 (non overlap)	12 (non overlap
Inhibit stack slave		28	20	20
Inhibit instruction slave			23	23
Inhibit operand slave	28		28	28
Inhibit full photo	31			
Lock translation slave rejection			25	25
Software Gen. System Error	6013	6013	4013 17	4013 17
Register Load ('Activate')	6014	6014	not implemented	not implemented
SMAC Image Store				
SMAC config. register (one per SMAC) (s = SMAC ident no, in range 0-F)	4C0s6A20	4C0s6A20	4Cs04A20	4Cs04A20
Available blocks	8-23	8-23	8-23	7-22
	bit 23=block 0, bit 22=block 1, bit 21=block 2, etc			bit 7=block 0, bit 8=block 1, etc
SMAC blocksize	128K /256	128K /256	128K	256K

2960.Bits SML16 2970.Bits 2976.Bits SML12 2980.Bits

SAC Trunk Register Image Store (2980 only)

DATA IN (B->A)

SPAD contents (i.e. RAM) { 40pt4000 0-31; 4100 32-63; 4200 64-95; 4300 96-127

SAC Registers

These are all model-independent.

[Footnotes:

1) 2980 Real-time Clock

pt is where you get the IPL interrupt from: p is normally 0 and t is "don't care" (zero).

You must also set the appropriate port bit in 4012 (bits 16 to 25 = Ports 0 to 5) to allow SAC external interrupts to reach the OCP.

The Real-time clock on the 2980 is out in a (the) SAC. ← Must also set bits 8-11 of S2 (I.S.4013) to port of clock to enable RATC to work

The 2976 is as the 2980 but p' = OCP port.

2) 2960 Real Addressing Mode

This is now thought to be available.]

J.K. Yarwood
P.D. Stephens

2970 SMAC IMAGE STORE MAP

I.S. ADDR	4CXX6004	4CXX6100	4CXX6A00	4CXX6A10	4CXX6A20
Data bit No.	Failing Data	Failing Address	Status	Engineering Status	Configuration
32	Hamming Parity 0	Address Parity		On Margins	0 BLK Conf as Ø
33	Hamming Parity 1			Inhibit Hamming Check	1 "
34	Hamming Parity 2			Inhibit Hamm. Report	2 "
35	Hamming Parity 3			On 'Engineers'	3 "
36	Hamming Parity 4	Address bit 04		Basic Unit Select	
37	Hamming Parity 5	Address bit 05		Add. Unit Select	
38	Hamming Parity 6	Address bit 06		Add. Unit Select	
39	Hamming Parity Word-	Address bit 07		Add. Unit Select	
40		Address bit 08	Port 0 Fail		Block 15 On line
41		Address bit 09	Port 1 Fail		Block 14 On Line
42		Address bit 10	Port 2 Fail		Block 13 On Line
43		Address bit 11	Port 3 Fail		Block 12 On Line
44		Address bit 12	Hamm. Gen. Fail		Block 11 On Line
45		Address bit 13	Hamm. Corr. Fail		Block 10 On Line
46		Address bit 14	I.S. NACK		Block 09 On Line
47		Address bit 15			Block 08 On Line
48	Byte 0 Parity Fail	Address bit 16	Byte Parity Fail		Block 07 On Line
49	Byte 1 Parity Fail	Address bit 17	Address Parity Fail		Block 06 On Line
50	Byte 2 Parity Fail	Address bit 18	Byte Func.Par.Fail		Block 05 On Line
51	Byte 3 Parity Fail	Address bit 19			Block 04 On Line
52	Byte 4 Parity Fail	Address bit 20			Block 03 On Line
53	Byte 5 Parity Fail	Address bit 21			Block 02 On Line
54	Byte 6 Parity Fail	Address bit 22			Block 01 On Line
55	Byte 7 Parity Fail	Address bit 23	Multibit Hamm. Fail		Block 00 On Line
56		Address bit 24	Singlebit Hamm. Fail	Marginate + 5v	Not Natural SMAC No.
57		Address bit 25	Reading Fault	Marginate - 5v	Be SMAC No. 1
58		Address bit 26	IPC 'X' Toggle Set	Marginate 19v	Port 0 Closed
59		Address bit 27	Holding Fault	Marginate 3.5v	Port 1 Closed
60		Address bit 28	Control Parity Fail	+5v Margins High	Port 2 Closed
61		Address bit 29	A.U.Addr.Par. Fail	-5v Margins High	Port 3 Closed
62				19v Margins High	BLK 1 in BU
63				3.5v Margins High	BLK Ø in BU

Image store locations on P2; subblock 60

All locations are read/write unless otherwise specified (or reserved)

6000	-	LSTB0
6001	-	LSTB1
6002	-	PSTB0
6003	-	PSTB1
6004	-	Reserved
6005	-	Reserved
6006	-	Handkeys (read only)
6007	-	Machine type (read only)
6008	-	Hooter (read zero, write bit 2**0 to set/clear hoot)
6009	-	Broadcast system errors (bit 2**0) (read and write)
600A	-	System interrupt mask
600B	-	Communications SMAC number
600C	-	RTCX
600D	-	RTCY
600E	-	RTCZ
600F	-	Clock module number
6010	-	Reserved
6011	-	CPU mode and control
6012	-	Reserved
6013	-	Cause system error (read zero, write to bit 2**7)
6014	-	Block 0 activate (write only)
6015	-	Reserved
6016	-	Upper lights (write only)
6017	-	Lower lights (write only)
6018	-	Stop machine (write only)

R.D. Eager
17th August 1982

Details of image store X6011 on P2

Image store read

A read returns bits 2**15 to 2**0 of the OCP control register (working store X27).

Image store write

The effect of a write is controlled by bits 2**16 and 2**17 of the operand. The combined values of these two bits have the following effect:

00 => Bits 2**0 to 2**15 of operand replace corresponding bits in the control register
01 => Bits 2**0 to 2**15 of operand are 'or'd into the control register
10 => Bits 2**0 to 2**15 of operand are cleared in the control register
11 => Illegal; gives program error 9, subclass 4

Meaning of bits in control register

2**0 - Inhibit full photograph
2**1 - Inhibit photograph
2**2 - Not used
2**3 - Inhibit data slaves
2**4 - Not used
2**5 - Inhibit retries
2**6 - Not used
2**7 - Inhibit hamming correction reporting
2**8 - Not used
2**9 - Pull HELP on program error
2**10 - Stop on program error
2**11 - Not used
2**12 - Inhibit photo on soft system error
2**13 - Retry in progress
2**14 - Stop after photograph
2**15 - Error handling in progress

R.D. Eager
16th August 1982

EMAS 2900 HARDWARE NOTE

No: 2
Date: 15/12/77

ICL 2970 Engineer's Panels

The attached description of the 2970 Engineer's panels has been taken from two ICL technical documents (diagrams: 65/94562, text: 65/94563); it describes the current state of the panels, i.e. following the SML19 hardware upgrade.

The question of providing this information was raised originally by my receipt (via John Maddock) of a similar document produced by the Post Office, relating to the Barbican machine. A number of people using the 2970 expressed an interest in it. However, although the P.O. document contains a slightly fuller description of the handkeys and indicators than the attached note, it is now out of date as a result of SML19. Nonetheless, much of the information in it is still relevant to our 2970; I have kept it and can give anyone interested a listing. It has been filed as Hardware Note 2a.

I should like to thank Les Mowbray (ICL 2900 Engineer) and John Maddock for their help.

John M. Murison

1 GENERAL LAYOUT

The OCP Engineer's facilities comprise 19 rows of monitor lamps with 72 lamps in each row & 1 row of monitor lamps with 36 lamps and 3 rows of handkeys with 36 in each row. The layout of handkeys and monitor lamps are shown on OD61 to 64 of document 65/94562 and their operational usage is described in the following text.

2 MONITOR LAMPS (Brackets at row number indicate MDP socket)

Row 1 (1R) The group of seven lamps at the left-hand end of this row display ARU Fails; AK, LX, AL, AQ, XC, AN and MF. The remaining lights display the ARU K highway (bits 00-63).

Row 2 (2R) The ARU Parity Flags are indicated by the group of six lamps situated at the left hand end of this row; AK, LX, AL, AQ, ASE (Inhibit System Error), and FP (Force Parity error). The remaining lamps display the ARU L highway (bits 00-63).

Row 3 (3R) The ARU AA and AP parity stats. (E, A and L) are displayed at the left hand end of this row. The remaining lamps are devoted to displaying the ARU Q highway (bits 2-63).

Row 4 (4R) The ARU AS and RAM parity stats. (E, A and L) are displayed at the left hand end of this row, the remainder display the contents of the ARU Ram (00-63).

Row 5 (11R)

RIF	00 - 06	IFU
RIK	00 - 06	
RHF	00 - 06	
RHK	00 - 06	
IL	00,01	
ISL	27,28	PC
RIF	27 - 30	
RHF	27 - 30	
FI		Function Dependant Flags
FSK		
FTOS		
FD		AGU Microprogram Flags
FX	General Purpose Flags	
FY		
FZ		
FTS		
JAM	Real Address Mode	
D	PSR D bit see IM110 for fuller explanation	
PRIV	Privilege	
DGW	Diagnostic Write	
MK	E,A,L Parity Stats	
RMFP	ARU Mask and Filler Parity	
RMF	00 - 15 ARU Mask and Filler.	

© International Computers Limited

Row 6 (12R)	RGF	00 - 06	AGU Function Register
	RGK	00 - 05	
	ROP	00, 14 - 31	
	IBE		Instruction Buffer empty.
	SUCJ		Jump condition satisfied.
	EP		Event pending
	RAN	P, 00 - 07	ARU AN register
	RAP	00 - 03, 62, 63	ARU AP bits
	RMK	30, 31, 62, 63	ARU MK bits
	MILLSH	OM, OL, 01-05	
	APSH	00-02	AP shift
	MK	$\overline{01}$, $\overline{02}$	
	AA	$\overline{01}$	
	RAG	P, 00-03	
Row 7 (13R)	PC	00-30	AGU PC
	BV		AGU B Overflow.
	CC	00,01	AGU CC
	GAZ	00,01	AGU ACC size.
	GOZ	00,01	AGU Operand size.
	GTWIST		AGU TWIST stat.
	RRF	00-07	Buffer Function Register.
	MPSLD		Loading mps (HWW7 or overlay)
	FMP		Microprogram loaded flag
	FHMP		Hardwired microprogram flag
	SPX	00,01	ARU Scratchpad ext. bits
	AT	00,01	ARU Arithmetic type.
	DS		ARU Double/Single length shifting.
	JF		ARU Jump Flag.
	JFA		ARU Jump Flag A
	SA		ARU Address SPAD from AN.
	RE		ARU Register Extension
	LF		ARU Link register from microprogram bits.
	SPOS		ARU Sign Positive (Dec. arithmetic).
	CY		ARU mill carry flag.
	PM		Program Mask (PSR)
	P+C	00,01	ARU Multiplier.
	CS		ARU Carry Save.

© International Computers Limited.

Row 8 (14R)	WA	00-31	AGU WA register.
	EMFV		Emulation Floating point overflow
	V		Common overflow
	CC	00,01	Common CC
	RAZ	00,01	Buffer ACC size
	ROZ	00,01	Buffer Operand size
	ARUTWIST		ARU TWIST Stat.
	RAF	00-06	ARU Function Register.
	FBOOTS		Hardwired Bootstrap.
	LOAD DEC	00-03	Direction bits.
	GMPS AD DEC	00-02	AGU Microprogram Address selection.
	MPS DATA	00-13	AGU MPS Data Buffer (see Row 9).
Row 9 (15R)	WB	00-31	AGU WB register.
	AV		ARU Overflow
	CC	00,01	ARU CC
	NEW AZ	00,01	ARU New Instruction ACC size.
	OLD AZ	00,01	ARU Previous Instruction ACC size.
	MPS DATA	14-44	AGU MPS Data Buffer.
Row 10 (16R)	SB	00-31	AGU SPAD output highway.
	SB	P0-P3	AGU SPAD output highway parity
	MPS CURR ADD.	00-11,P	AGU Microprogram Current Add.
	MPS ACC ADD.	00-11,P	AGU Microprogram Access Add.
	NULL		Null Beat (AGU)
	FADD		Forced Address (AGU).
	FWD		
	STOLK3		AGU Lockout.
	LKS1		AGU Lockout.
	LKS2		AGU Lockout.
	ADD LKS		"
	RUN LKS		"
	STL1		"
	STL2		"
Row 11 (17R)	K	00-31	AGU K highway.
	DT	00,01	Descriptor Type.
	DZ	00-02	Descriptor Size.
	A		A bit.
	USC		Unscaled.
	BCI		Bound Check Inhibit.
	FRAM	00-06	Function RAM Addressing.

© International Computers Limited

	KRAM	00-06	K Field RAM Addressing.
	IRAM	00-06	Descriptor RAM Addressing.
	vi Access	00-02	Volatile Index Address Selection.
	VI ADD	00-07	Volatile Index Addressing.
Row 12 (18R)	L	00-31	$\overline{\text{AGU L}}$ highway.
	FSP 2 & 3		AGU SPAD Flags.
	SPAD ADD	00-03	" " Addressing
	OTB		Valid Stats. (Microprogram Address valid for replaced jumps if required).
	GOB		
	DTB		
	DZB		
	INTB		
	RLOTB		
	REALER		
	ARUSTP		
	AMP AD DEC	00-02	ARU Microprogram Address selection.
	REPLMT		
	FPNXT		Next ARU func. is floating point.
	RFB		AGU/ARU Interlocks.
	RBB		" "
	AAR		" "
	OPINT		Operand in Internal Format.
	ACINT		ACC in Internal Format.
	AOK		
	BUF 2F		Association Address
	SSNEQ		Stack Segment Number Equivalence
	ISL		Instruction Slave — Slave hit
	SSL		Stack slave — Slave hit
	IB		Slave last hit
	GBM		Slave last hit
	GBL		Slave last hit
	RBM		Slave last hit
	RBL		Slave last hit
	LKS 1		ARU Lockout.
	LKS 2		"
	PLKS		Pipe Locks

© International Computers Limited

Row 13 (19R)	Q	00-31	AGU Q highway.
	FSP0 & 1		AGU SPAD Flags
	MPS	00-35	ARU Microprogram word.
Row 14 (20R)	GBM	00-31, 0P-3P	AGU GB Most.
	MPS CURR ADD	00-11, P	ARU Microprogram current address.
	MPS ACC. ADD	00-11, P	ARU Microprogram Access address.
	WFL		ARU Wait for Literal.
	FADD		ARU Forced Address.
	VIADD	00-06	ARU Volatile Index Address.
Row 15 (21R)	GBL	32-63	AGU GB Least.
	GBL	04-07	AGU GB Least Parity
	TA	04-29, P	Store Address, (Real Address).
	BYTE FUNC	00-07, P	
Row 16 (22R)	PORT 0 SEINT		Direct wired interface interrupts.
	PORT 0 EXINT		
	PORT 0 OPINT		
	PORT 1 SEINT		
	PORT 1 EXINT		
	PORT 1 OPINT		
	PORT 2 SEINT		
	PORT 2 EXINT		
	PORT 2 OPINT		
	PORT 3 SEINT		
	PORT 3 EXINT		
	PORT 3 OPINT		
	MVSI		Masked Virtual Store Interrupt.
	MPE		Masked Program Error.
	MSCI		Masked System Call Interrupt.
	MOUT		Masked Out Interrupt.
	MXC		Masked Extracode.
	RGN	24-31	AGU $\overline{GN}$
	RGNZR		AGU GN bits 8-23 = zero.

© International Computers Limited

Row 16
Cont.

ACCESS E		Store Access Execute.
ACCESS W		Store Access Write.
ACCESS R		Store Access Read.
APF	03-11	Access Protection Field bits.
ACR	00-03	Access Control Register.
PERF		Permission Fail.
MEQ		Multiequivalence.
TAC		ATU Accept (Hit).
UMS		Update Miss (Use bits).
MS		Miss.
MISS		
PAGEI		Paged segment address.
VAM		Virtual Address Mode.
SIP		Search In Progress (TS).
LOAD VIR		Load Virtual Address.
LOAD PAP		Load Page and Permission bits.
LOAD RRE		Load Real Even address.
LOAD REO		Load Real Odd address.
WD M, L		
CPR	P0-P2	ATU CPR Pointer.

Row 17 (23R)	SSE	Interrupt Register
	GPE	
	VSI	
	EP	
	OUT	
	XC	
	SE	
	SGSE	
	RTC	
	XACT	
	HRI	
	APE	
	IT	
	IC	

© International Computers Limited

Row 17 Cont.	FKI	Function K and Indirect RAMS	Parity Fail Stats.
	ASE	ARU System Error	
	IB	Instruction Buffer	
	RC	ARU control	
	GC	AGU control	
	GK	K highway	
	GL	L highway	
	GQ	Q highway	
	GCY	Carries	
	GMISC	AGU Misc.	
	SSD	Stack Slave Data	
	SSA	Stack Slave Address	
	ISD	Instruction Slave Data .	
	ISA	Instruction Slave Address	
	SMAC		
	SDIE	Software DIE	
	DIE	Hardware DIE	
	TRANS		
	CHECK		
	MODE		
	PERMIT		
	BUSY	Current stats. of ATU	
	RIP		
	REQ		
	APHIP	ARU Photograph in Progress	
	CASS	Cassette is Busy	
	FEM	Emulation mode flag.	
	EM 0-3	Emulation mode.	
	H BUSY	Hash Busy	
	STREQ	Store Request	Requests from input device
	MPREQ	Microprogram request	to main and Microprogram store.
	OFFLINE		
	PERI	Perifail	Diagnostic Unit
	STORE	Store Fail	Input Device
	UNSERVED		Fails
	CNT	Count Fail	
	MPSINH	MPS Inhibit Fai.	
	RAM Add. X0-3	ATU RAM Address.	
	Y0-3		
	Y01,Y23	ACR use bit checks RAM address since only one ACR use bit is held for each Virtual Address Pair (even and odd).	

© International Computers Limited

Row 18 (24R)	WD	00-63, P0-P7	Write Buffer store.

Row 19 (25R)	ARU COPY BUFFER	00 - 11		
	AGU COPY BUFFER	00 - 11		
	1900 EM LINK	00 - 11	1900 Emulation Link (AGU)	
	AGU SR	00 - 11	AGU SUBROUTINE	
	AGU LR	00 - 11	AGU Link	
	INHAM	Inhibit Hamming Interrupt		CW STATS
	INSTO	Inhibit Main Store		
	IRTY	Inhibit Retry		
	ISRS	Inhibit Successful Retry Reporting		
	ISTSL	Inhibit Stack Slave		
	IINSL	Inhibit Instruction Slave		
	ICPRS	Inhibit Current page/segment Registers		
	IPHOT	Inhibit Photo		

External Monitor Strip

OPTINT	0,1	Peripheral Interrupt
IDLE		
OPERAND	14-31	
HMP		Hard Microprogram
GMPADD	8-11	AGU Microprogram Address
SEINT	0,1	System Error Interrupt
SMACFL		SMAC Fail

© International Computers Limited

HANDKEYS

The handkeys are arranged in three rows A, B and C and are depicted on OD61 of document 65/94562.

Row A Handkeys 1 - 32 are used for setting up Image Store data.

Handkeys 33 - 36 vary the OCP Timing Margins as follows:

a) Key 33 AGU FAST down - All AGU timing speeded up by 5%
b) Key 34 AGU SLOW down - All AGU timing slowed down by 20-25%
c) Key 35 ARU FAST down - All ARU timing speeded up by 5%
d) Key 36 ARU SLOW down - All ARU timing slowed down by 20-25%

Note: The results of setting AGU FAST and AGU SLOW or ARU FAST and ARU SLOW is indeterminate.

e) All four switches in centre position - timing at nominal speed.
f) Key 33 up - Individual AGU timing pulses as determined by a decode of the remaining 3 switches speeded up by a constant 5 n Sec (see table 1).

Switches			AGU Timing Pulse
KEY 34	KEY 35	KEY 36	Speeded up by 5n Sec.
Centre	Centre	Centre	T08TM
Centre	Centre	TDEC2	T14TM
Centre	TDEC1	Centre	T17TM
Centre	TDEC1	TDEC2	T19TM
TDEC0	Centre	Centre	T12DF
TDEC0	Centre	TDEC2	T12TM
TDEC0	TDEC1	Centre	T17GMIL
TDEC0	TDEC1	TDEC2	Conditional Jump

Table 1. AGU timing Pulse Margin Speed Up

g) Key 33 centre - Individual ARU timing pulses as determined by a decode of the remaining 3 switches speeded up by a constant 5n Sec (see table 2).

© International Computers Limited

Switches			ARU Timing Pulse
KEY 34	KEY 35	KEY 36	Speeded up by 5n Sec.
Centre	Centre	Centre	All AGU and ARU Timing Nominal
Centre	Centre	TDEC3	T11DF
Centre	TDEC1	Centre	T10TM
Centre	TDEC1	TDEC3	T08RTIM
TDEC0	Centre	Centre	T12TIM
TDEC0	Centre	TDEC3	T12RTIM
TDEC0	TDEC1	Centre	T17RTIM
TDEC0	TDEC1	TDEC3	T17MILL

Table 2. ARU Timing Pulse Margin Speed Up

© International Computers Limited

Row B This row of keys enables the OCP to stop on a specified address, stopping on either microprogram equivalence (handkeys 1 - 12) or, on PC and ZA equivalences (keys 1 - 31). The specific equivalence on which the OCP stops is determined by the four keys situated at the extreme right-hand end of this row (keys 33 - 36).

MISC STOP, Key 32 down, stops the OCP when the AGU Miscellaneous Function 'STOP ON MISC' is set.

Stop on Microprogram loaded/PC; Key 33 enables the OCP to be stopped when the microprogram has completed loading by detecting #F (Stop character) in the direction bits. The other position of the key "Stop on PC equivalence" stops the OCP immediately equivalence is detected between the bits set-up on the handkeys and PC providing that "Stop on AGU/ARU Microprogram address" is not set. The OCP will resume running when the key is returned to the centre off position.

Stop on - ARU/AGU microprogram address; Key 34 enables the OCP to be stopped when bits 00-11 of the ARU/AGU microprogram address are equal to the bits set up on the keys. The OCP will resume running when the handkey is returned to its centre-off position.

When both the "Stop on PC" Keys 13-31 and "Stop on AGU microprogram address" 1-12 are set, then the OCP will only stop when equivalence is detected between these conditions and that set up on the keys.

ZA Read/ZA All and ZA Write/ZA Execute; Keys 35 and 36 enable the OCP to be stopped when ZA (which normally contains the virtual address) is equal to the bits set on the keys. The OCP will resume running when the appropriate key is returned to its centre-off position.

Keys 1 - 12 can also be used to set up the Port, Trunk and Stream when doing a Bootstrap or dump (Row C handkey 34).

FEM, Key 13 down, stops on selected address (Keys 1 - 12) when in emulation mode and stops on selected address in normal mode when key in neutral position.

A further facility exists on this row in that the group of keys at the left-hand end can be set to force either the ARU K (AK 00-02) L (AL 00-02) and KX (AKX 00-02) highways and the ARU scratchpad (ASP 00-03) and AC→AK, or alternatively the AGU Scratchpad Flags (0-3) and AGU Scratchpad, and J Highway. The condition to be displayed is dependant upon the setting of the Display ARU/ Display AGU handkey (key 31 Row C).

© International Computers Limited

Row C

KEYS

1	– ENG	Enables Engineers panel and disables OPER
2	FORCE IT INHIBIT IT	Force Interval Timer interrupt Inhibits interrupts from the Interval Timer and RTC
3	RAM ACR0	Force Real Address Mode Force ACR0
4	STOP ON SMAC FL IGNORE SMAC FL	Allows OCP to stop on unrecovered SMAC Fails Allows OCP to ignore unrecovered SMAC Fails
5	STOP ON SE FL IGNORE SE FL	Stops OCP on System Error Fail Allows OCP to ignore System Error Fail
6	INHIBIT CPR EX INHIBIT CPR ALL	Prevents the CPR's being used during execute accesses Prevents the CPR's being used during all store accesses
7	– INHIBIT PORT ACT	Prevents the OCP from taking any action as a result of receiving an ACTIVATE
8	– INHIBIT PORT INT	Prevents the OCP from raising an interrupt as a result of receiving a port interrupt.
9	– STOP ON SACTOUT	Causes OCP to stop on SAC Time out
10	– INHIBIT IC	Inhibits the instruction counter interrupts
11	STOP ON TIMEOUT IGNORE TIME OUT	Causes OCP to stop on any time out Allows OCP to ignore any time out

© International Computers Limited

KEYS

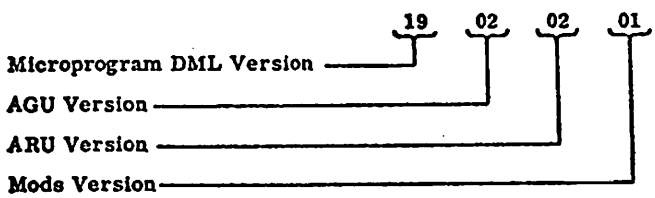
12	RESET FAILS INHIBIT PHOTO	Resets any outstanding fail Allows the system Error Interrupt routine to omit the Photo of the OCP hardware
13	— INHIBIT INSTSL	Inhibits the Instruction slave
14	RESET LOCKOUTS INHIBIT INTLKS	Reset any Lock outs (or any timing interlocks) Allows AGU and ARU to run irrespective of any interlocks that may be set.
15	— INHIBIT STK SL	Inhibits the Stack Slave causing all read accesses to come from main store.
16	— INHIBIT HAM INT	Inhibits any Hamming Interrupt
17	INHIBIT SRR INHIBIT RETRY	Inhibits successful Retry reporting Allows the system Error Interrupt to by pass the hardware retry.
18	— ENABLE SMAC FRZ	This key allows the OCP general reset (FREEZE) to be passed via a Port to the SMAC.
19	DUMP —	Results in a PAW Function of Dump being sent to the Controller selected on the Keys of Row B when the RUN Key (35 Row C) is set OCP then stops.
20	— STOP ALL	Allows Stop on nulls (AGU and ARU)
22	STOP AGU AGU LOOP	Stop AGU Clocks. Inhibits the strobing of the AGU microprogram data buffers, thus enabling a specific beat to be repeated indefinitely.
23	STOP ARU ARU LOOP	Stop ARU clocks. Inhibits the strobing of the ARU microprogram data buffers thus enabling a specific beat to be repeated indefinitely.

© International Computers Limited

25	— EKS	This key is the M.S.B of the Engineers Facility Field and also the drop through Idle
26-29	EF00 01 02 03	These keys have two functions:- (X) They are looked at, at Microprogram load time (IPL), and if the decode is #5, the VAR Register is loaded from the Image Store Keys Row A. (Y) These four Keys and EKS (Key 25) provide a means of forcing several functions manually provided the Engineers facility (Key 32 Row C) is operated. The OCP should be at Real OTB. Key 35 Row C should be in OI position. The Engineers facility is performed by setting the decode, then pressing and releasing the Engineers Facility Key (Key 32 Row C), followed by the operation of the GO Key (Key 36 Row C). The facilities provided are as follows:-

Engineers Facilities.

0		No Action
1	PC JUMP	Jump to Address on Image Store data keys Row A as per address mode, Virtual or Real.
2	MPVERS	Microprogram Version displays on the AGU Register WB display on Row 9. The decode is Hexadecimal. For example:



3-7		No Action
8	READ	Read from virtual address in Store. The address is set on Image Store Keys Row A. The information read is displayed in Registers GBM and GBL Rows 14 and 15.

© International Computers Limited

9	READ +	Read from virtual address is store. The address is set on Image Store Keys Row A. The information read is displayed in Registers GBM and GBL Rows 14 and 15. EKS key being pressed and released increments the virtual address by 2 words and the new information is displayed in GBM and GBL Registers. The microprogram can be returned to real OTB by returning EF 00-03 keys to neutral and pressing and releasing EKS Key.
A	Read R	As 8 (READ) but the Real Address Mode Key (Key 3 up Row C) operated.
B	Read R +	As 9 (READ +) but the Real Address Mode Key (Key 3 up Row C) operated.
C	WRITE	Write to the virtual address set on the Image Store Keys Row A upon the operation of the GO Key (Key 36 Row C). The machine will now wait for data to be set on the Image Store Keys. When EKS Key is operated this data is written into the virtual address in store
D	WRITE +	Write to the virtual address set on the Image Store Keys Row A upon the operation of the GO Key (Key 36 Row C). The machine will now wait for data to be set on the Image Store Keys. When EKS key is operated this data is written into the virtual address in store. Further press and releases of the EKS key increment the address by 2 words and write the data on Image Store Keys. The microprogram can be returned to real OTB by returning EF00-03 keys to neutral and pressing and releasing the EKS key.
E	WRITE R	As C but Real Address Mode key operated (Key 3 up Row C).
F	WRITE R+	As D but Real Address Mode key operated. (Key 3 up Row C).
10	MOVEOVER	Move overlays from existing to new area of store set on Image Store keys (Row A).
11	FOTO	Force Photo, done automatically by this key.
12	READ INST SLAVE	Read the line of the Instruction Slave defined on Image Store Keys, and appears on monitor lights GBM and GBL Rows 14 and 15.
13	READ STACK SLAVE	Read the line of the Stack Slave defined on Image Store Keys, and appears on monitor lights GBM and GBL. Rows 14 and 15.
14,15		No Action.

© International Computers Limited

16 READ AND WRITE SAC AND SMAC I.S.

To read set the Engineers Facility keys, set the image store address as defined below *, and perform the functions described in paragraph (Y), preceding the Engineers Facilities. The information in GBM is the Image Store address which has been set up and in GBL is the information Read.

To Write set up Image Store Keys Row A, perform the Engineers Facility operation described in paragraph (Y) but after the GO key has been pressed the OCP will wait for data which is to be set up on Image Store Keys. EKS key is pressed and released this causes the Image Store write action to be completed.

* Image Store Keys Row A.

Bit 0 :- 0 Read, 1 Write

4,5:- Both down :- SMAC; 5 only :- SAC; No keys operated:- SAC Trunk.

SMAC
- Bits 8-15 SMAC number
- Bits 16-31 SMAC Image Store Address (see N.R.S.D 4,2,3 G)

SAC
- Bits 8-11 Port number of SAC
- Bits 16-31 The Image Store Address

17 No Action

18 WRITE SPADS

On Image Store keys 24-27 put the S.P. Flags, and on keys 28-31 put the S.P. line number. The Image Store Engineers Facility action described in paragraph (Y) is performed. The data to be written to the Scratch Pad is set on the Image Store Keys. EKS key is then pressed and released.

19 UPDATE SP

This updates the visible Registers to the current machine state automatically

e.g. ACS, CC, OV in PSR

SP SA (IT Copy)
SP SW (A0 Copy)
SP SX (A1 Copy)
SP SY (A2 Copy)
SP SZ (A3 Copy)
SP PC
SP SD (IC Copy)

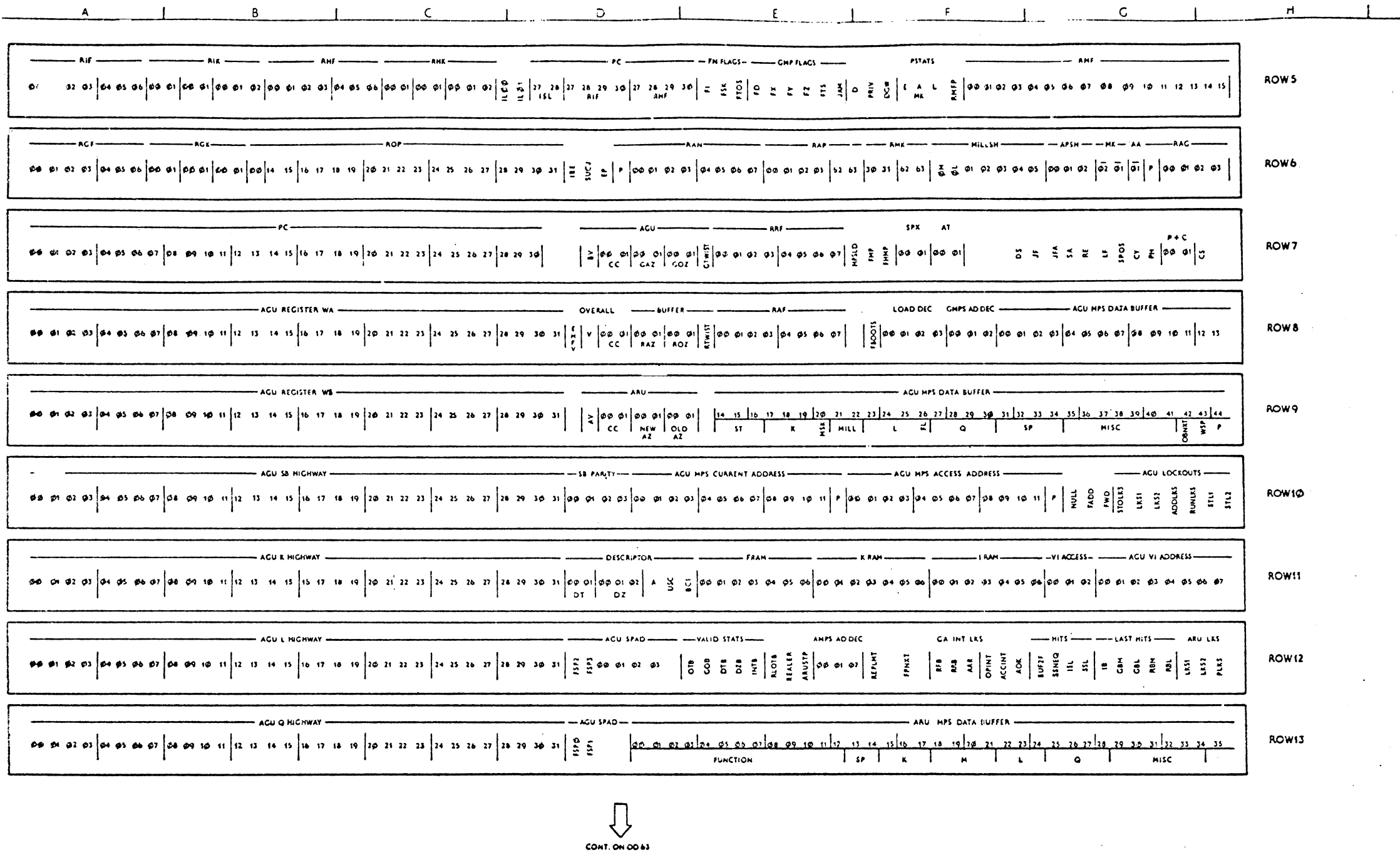
© International Computers Limited

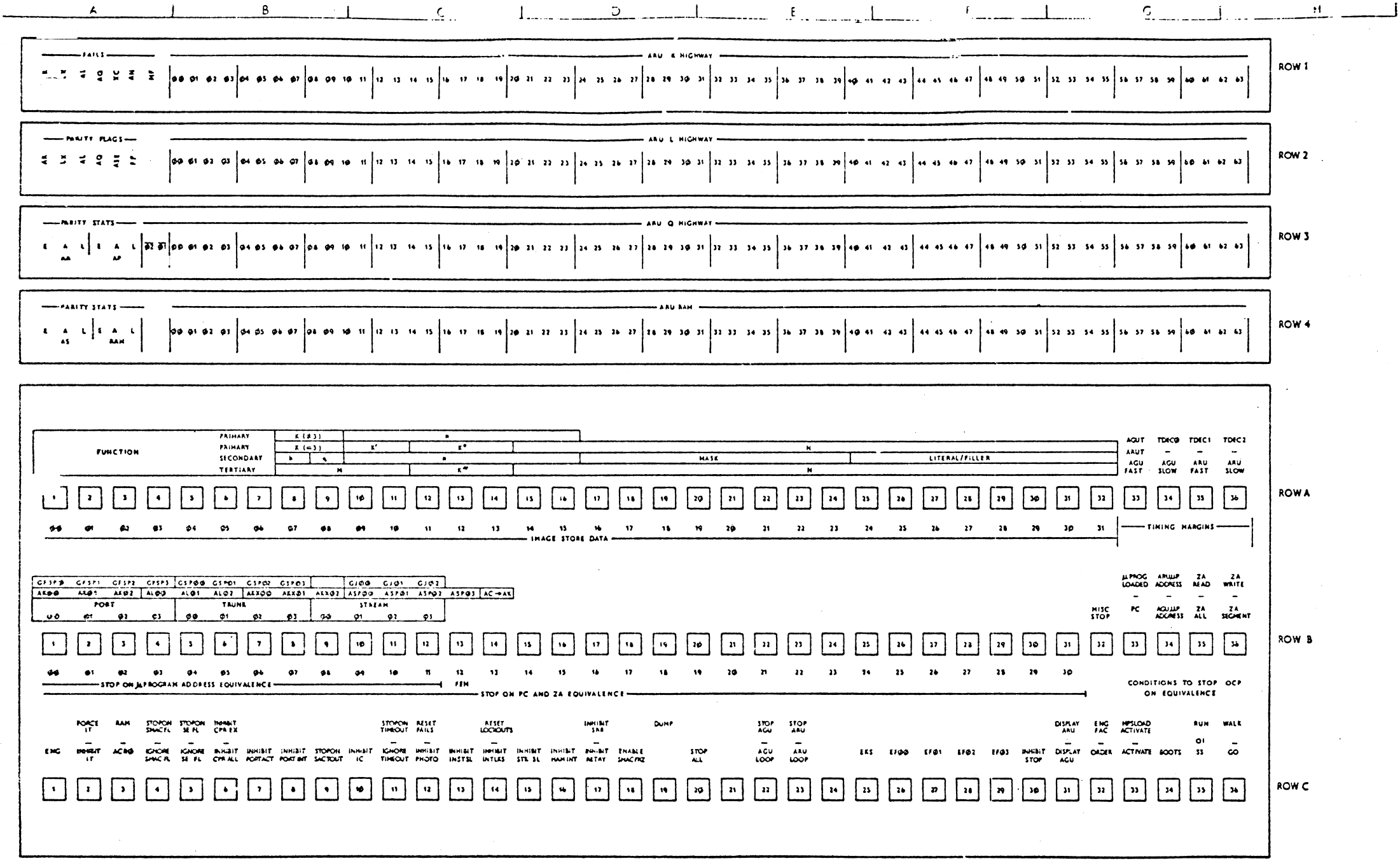
30	INHIBIT STOP	Used in conjunction with stop keys to provide Engineers Scope trigger.
31	DISPLAY ARU DISPLAY AGU	This key causes the Register(s) selected by the keys on row B to be placed on the highways which in turn are displayed on monitor lamps.
32	ENG FAC ORDER	Enables the Five Keys 25-29. The ORDER position allows the primitive interface order set up on ROW A to be obeyed. If only one order is required the RUN key (35 Row C) is placed in the appropriate position and the GO Key is pressed once.
33	MPSLOAD ACTIVATE	When operating the MICROPROGRAM LOAD switch it is assumed that the microprogram is in the main store starting at #100 BYTES, the RUN switch (Key 35) should also be set. The microprogram is then loaded and entered and if it is a normal Microprogram it carries out a register load from #80 BYTES etc.
33	ACTIVATE	The ACT position of this switch causes an entry to the microprogram such that if a normal microprogram is in, a register load from #80 BYTES etc. is carried out when RUN is set.
34	BOOTS	Operation of this key results in a PAW function of IPL being transmitted to the controller selected by the Keys of Row B. On receiving an interrupt from the controller the OCP loads its microprogram etc.
35	RUN	The RUN position allows the clock to run until it is stopped by a stop condition or RUN is switched off. The conditions for stopping are various and are as follows. 1. The stop conditions described under ROW B 2. System error with stop on System Error or system Error mask set. 3. Inhibit time out and the OCP is waiting for something. 4. AGU and ARU only with the interlock functioning.
35	OI	One instruction allows a single instruction for each operation of WALK/GO key.
35	SS	Single shot allows a single microprogram instruction for each operation of WALK/GO key.

© International Computers Limited

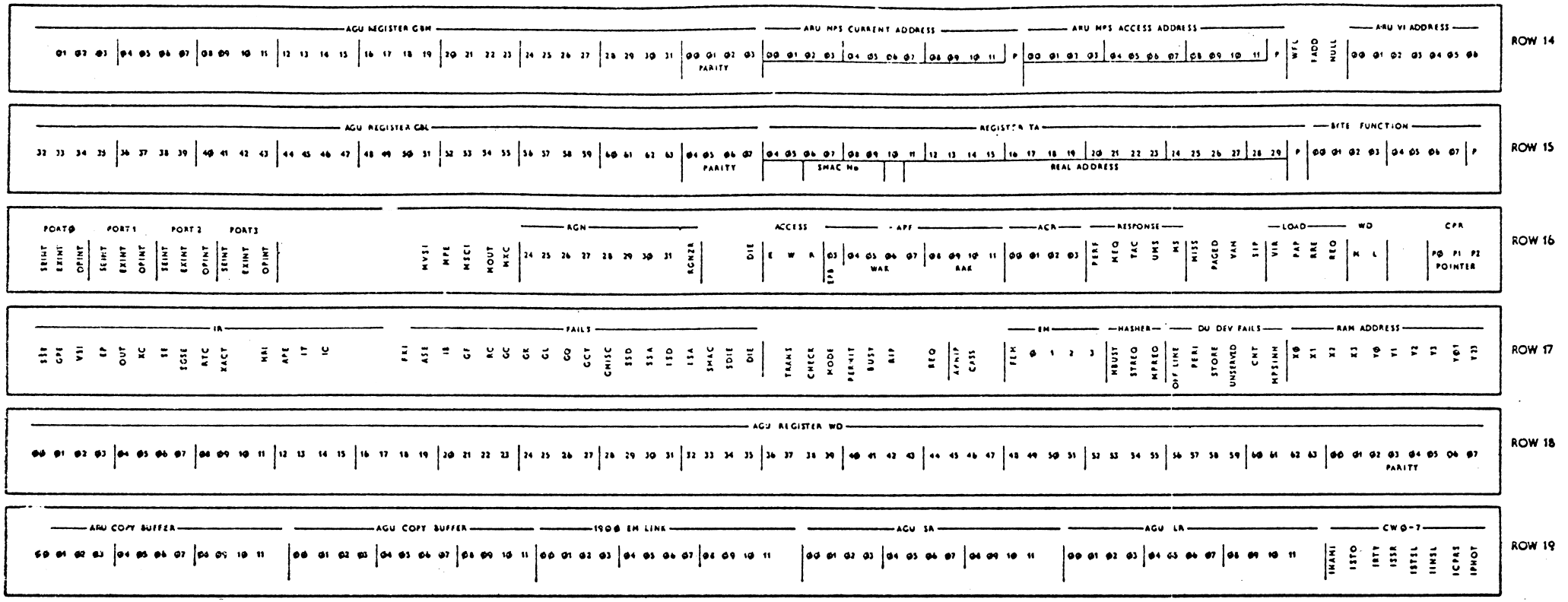
36	WALK	A 16 μ sec clock used in conjunction with OI and SS.
	GO	A Single Shot used in conjunction with OI and SS.

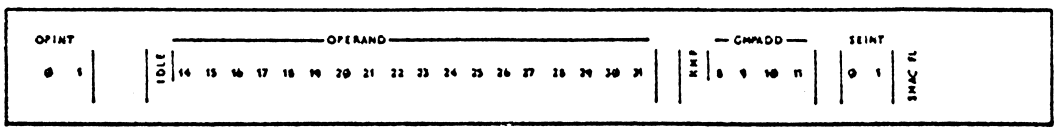
© International Computers Limited


A
B
C
D
E
F
G
H
ROW5
RIF
RIK
RHF
RHK
PC
FN FLAGS
GMP FLAGS
PSTATS
AMF
ROW6
RGF
RGK
ROP
RAN
RAP
AMK
MILLSH
APSH
MK
AA
RAG
ROW7
PC
AGU
RRF
SPX
AT
P+C
ROW8
AGU REGISTER WA
OVERALL
BUFFER
RAF
LOAD DEC
GMPS ADDEC
AGU MPS DATA BUFFER
ROW9
AGU REGISTER WB
ARU
AGU MPS DATA BUFFER
ST
R
MSK
MILL
L
FL
Q
SP
MISC
OBNXT
WSP
P
ROW10
AGU SB HIGHWAY
SB PARITY
AGU MPS CURRENT ADDRESS
AGU MPS ACCESS ADDRESS
AGU LOCKOUTS
ROW11
AGU K HIGHWAY
DESCRIPTOR
FRAM
K RAM
I RAM
VI ACCESS
AGU VI ADDRESS
ROW12
AGU L HIGHWAY
AGU SPAD
VALID STATS
AMPS ADDEC
GA INT LKS
HITS
LAST HITS
ARU LKS
ROW13
AGU Q HIGHWAY
AGU SPAD
ARU MPS DATA BUFFER
FUNCTION
SP
K
M
L
Q
MISC
CONT. ON 00 63



NOTE:
FOR CONNECTIONS BETWEEN M
AND BETWEEN MONITOR DIS
FOR SIGNALS FROM WANDERI





EXTERNAL MONITOR STRIP

EMAS 2900 HARDWARE NOTE

No: 3
Date: 13/12/79

How to Force a Program Error Interrupt from the Handkeys

Before a dump can be usefully taken the machine must be at IDLE "DEAD". This state is reached after most hardware and software errors. However, if the machine is not at IDLE "DEAD" an error should be forced, as follows:

For the 2960 (SM11, using TLOE/8):

On the ECP, type:

```
DW 9FFC
WMA 0 E
GO
```

For the 2970:

1. Off "RUN"
2. "ENG" down
3. Handkeys (top row) to ZERO
4. One tick on "GO" (usually, but more ticks may be necessary) until the RLOTB light comes on
5. "ORDER" down
6. Press "GO" until no lights on in RGF (2nd row of lights from top)
7. "ORDER" central
8. "RUN" - machine goes to IDLE "DEAD"

For the 2980:

1. Set switches as follows:

SMGO	UP	(Row 1 Key 1)
S.INST.MODE followed by S.SHOT MODE	DOWN/UP	(Row 6 Key 2)
I1 STOP	DOWN	(Row 3 Key 7)
STOP ON FAIL	UP	(Row 5 Key 4)
INH.ORD.STOP	DOWN	(Row 5 Key 9)

   These are the only switches that should be set, apart from the deck TRUNK/STREAM switches. Unset any others.

2. Turn to Frame 01 (Critic and Modifier Dataflow) on monitor display.
3. Set Serial Monitoring address keys to Frame 1, Line 3.
4. Set bit 17 on DATA-KOD keys.

5. Press SET REG (Row 1, Key 4 - UP). Check that lamp FSEI on Line 3 (bit 17) comes on.

6. Turn to Frame 10 (Modification and B Arithmetic Stage Microprogram) on monitor display.

7. Set RUN (Row 6, Key 2 - CENTRAL).

8. Press GO (Row 6, Key 1 - DOWN). Check that machine stops with I1 (Line 7, bit 4) lamp lit.

9. Turn to Frame 21 (SAC trunk links E, F, G, H) on monitor display.

10. Rest I1 STOP (Row 3, Key 7 - CENTRAL).

11. Press GO (Row 6, Key 1 - DOWN) - machine goes to IDLE "DEAD".

R.D. Eager
J.F. Livingstone
P.D. Stephens
J.K. Yarwood

EMAS 2900 HARDWARE NOTE

No: 4
Date: 19/12/77

Hardware Testing

A process, ENGINR, exists to be used for running hardware tests. Currently two tests exist:

a) A test of the VMY instruction accessing the B register and data which goes across a page boundary. This used to give an erroneous bound check on the 2970 when a page fault occurred during its execution; it has now been corrected. The test is preserved for use on other 2900 machines. To run it, type:

```
COMMAND: RUN VMYBY

DO YOU WANT SECOND PAGE IN CORE?
Y/N: N or Y

VMY B WORKED OK.
```

Note that the program should work whether or not the second page is in core. If the fault exists the program will work when the reply is 'Y', but will fail with a band check if the reply is 'N'.

b) RAND READ

This is a program based on RAND READ on System 4 EMAS. It can be used to test up to 4 disk drives. The disks are referred to by their FSYS numbers - e.g. PETE01 is disk 1, EMAS10 is disk 10.

The program is run thus:

```
COMMAND· RANDREAD

NO OF DISKS:          reply 1-4

FSYS NOS. OF DISKS:   reply with FSYS numbers

Q SIZE PER DISK:      reply 1-16, specifying number of transfers to
                      be PONned to each disk at once

READS/DISK:           reply with number of READs to be done
```

When the specified number of reads have been completed, the number of faults, cumulative since the start of the command, are reported and the user is then invited to continue. The reply should be Y to continue or N to stop the tests. If the user replies with Y, the READS/DISK: prompt is given again; the other parameters are held at their previous values.

Roderick McLeod

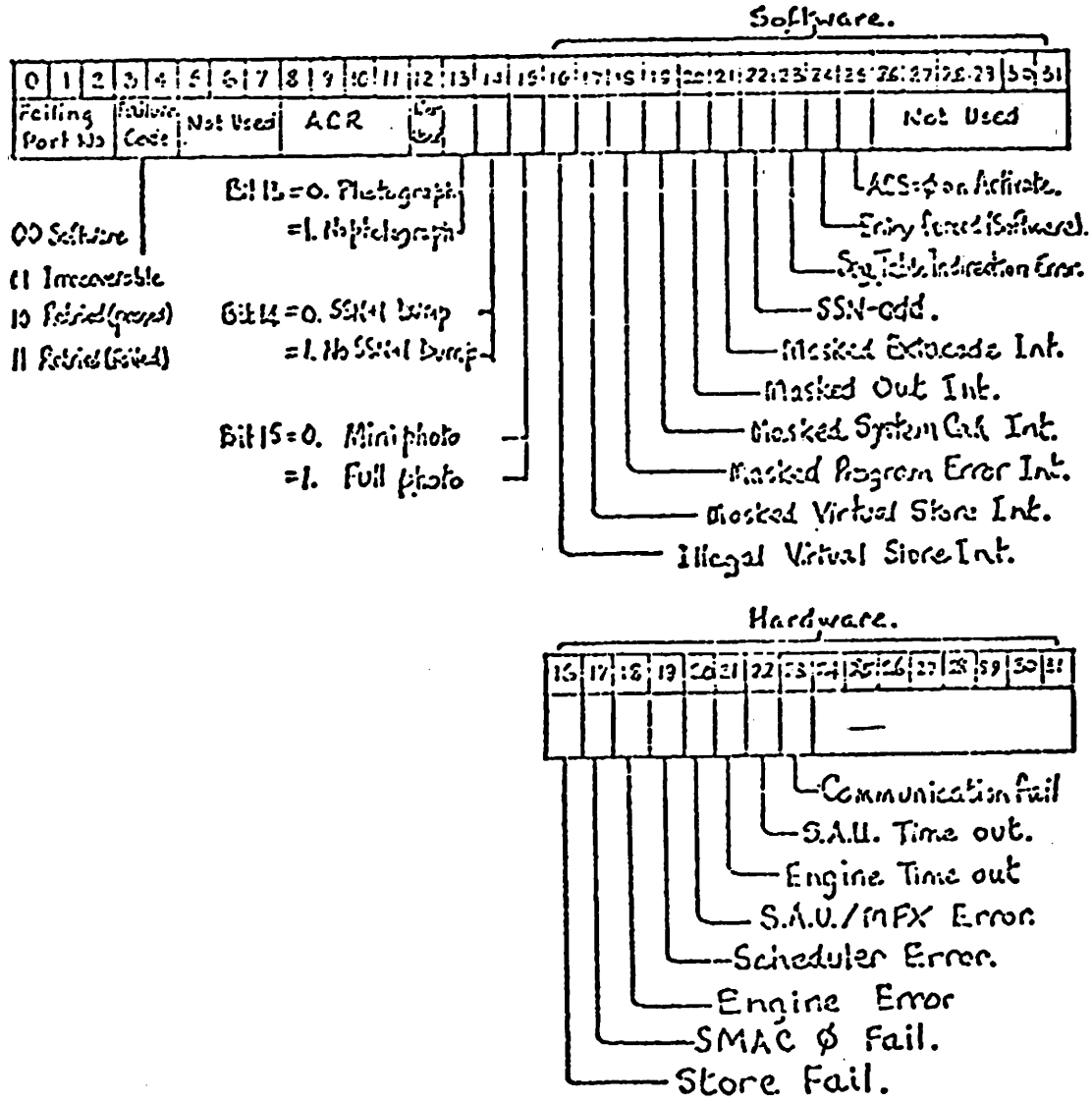
System Error Interrupt Parameters for 2960, 2970, 2980

This note is taken from a badly-Xeroxed copy of ICL document SYSB-22-0233.

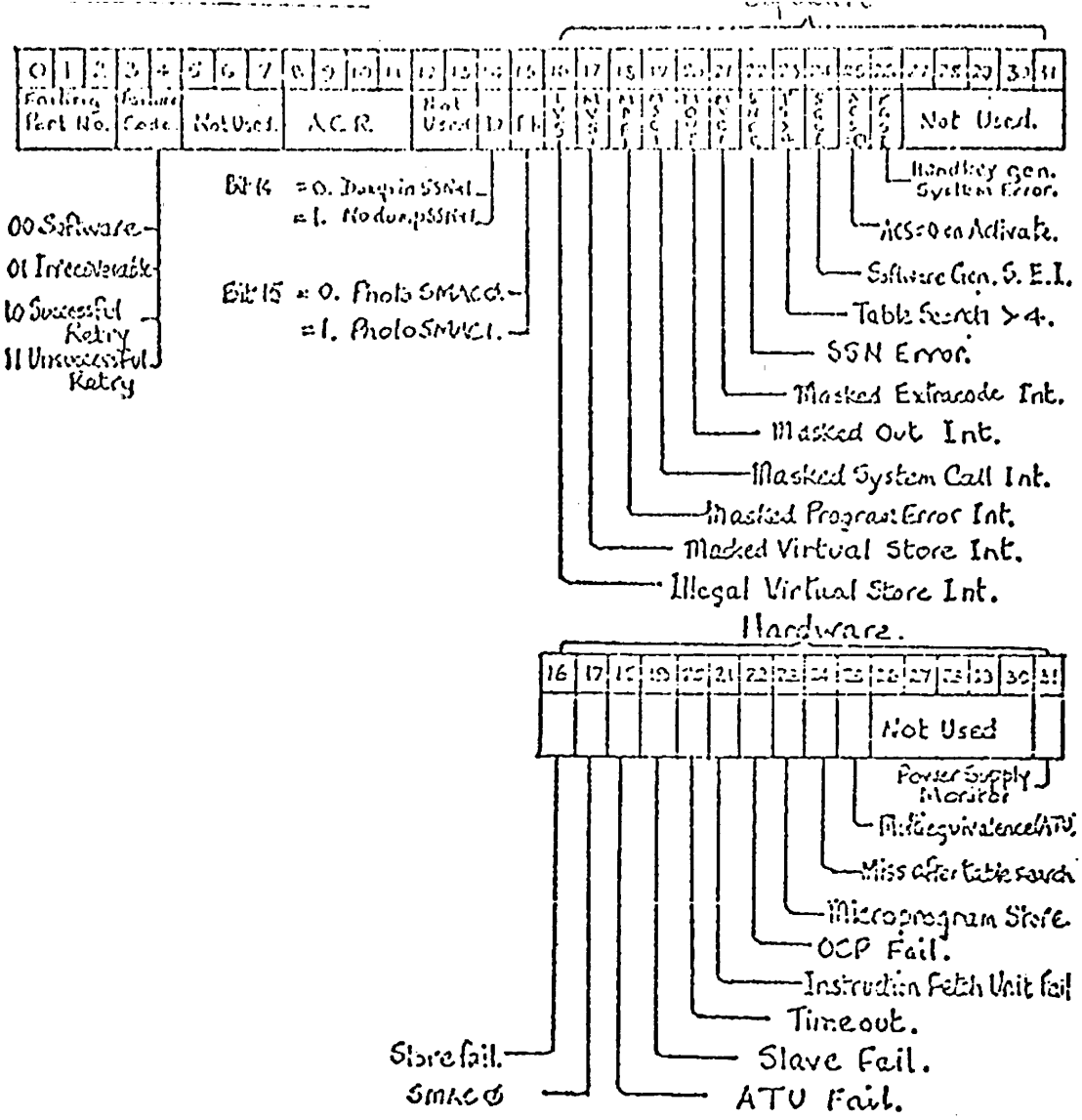
The System Error Interrupt Stack

The system error is a stack switching interrupt and the code of SEI Analyser is entered with its LNB at the stack base. Word 0 contains the stack segment number (in the current implementation this is public segment 8198). Words 1 and 2 are zero and words 3 and 4 point to the Procedure Linkage Table (PLT) of the compilation module containing SEI Analyser. Word 5 contains the SEI parameter as stacked by the hardware interrupt routine and word 6 is the segment number of the interrupted stack.

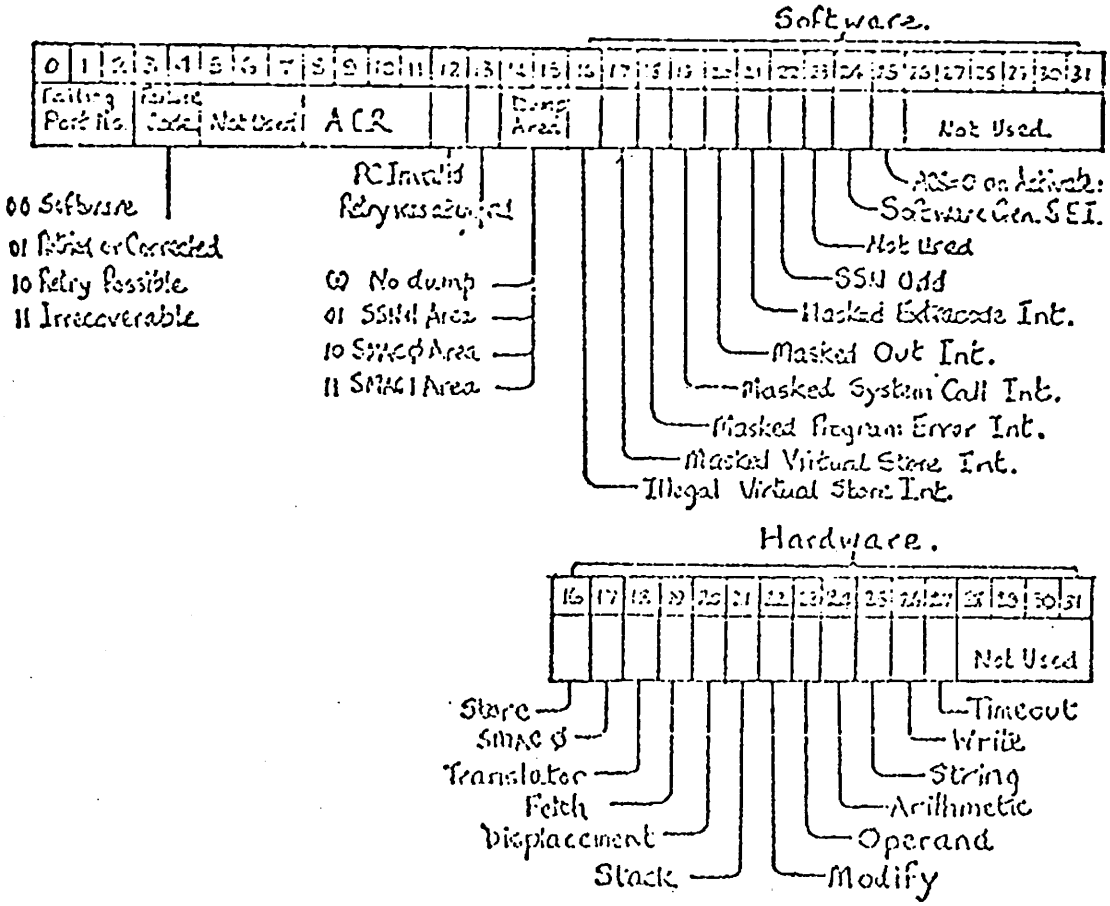
SEI Parameter for 2960



The parameter for interrupts other than on the current OCP port will look like an empty software failure parameter (i.e. bits 16-31 not used).



SEI Parameter for 2980



3.6 INTERRUPT PARAMETERS 2980

PARAMETER STACKED ON SYSTEM ERROR (n = 1)

Bit Nos	MEANING	
0 to 2	Failing Module No.	
3, 4	Failure code (xy)	= 00 Software SE
		= 01 Retried or Corrected
		= 10 Retry is possible
		= 11 Irrecoverable Failure
5 - 7	Undefined	
8 - 11	ACR	
12	PC invalid	
13	Retry was attempted	
14, 15	Dump Area	= 00 No Dump
		= 01 SSN + 1 area
		= 10 SMAC 0 area) See Section 8
		= 11 SMAC 1 area)

	Hardware System error	Software System error
16	Store	Illegal Virtual Store Condition
17	SMAC 0	Masked Interrupt : Class 5
18	Translator	Class 7
19	Fetch	Class 8
20	Displacement	Class 9
21	Stack	Class 10
22	Modify	SSN is odd number
23	Operand	Segment Table Format Error
24	Arithmetic	Entry forced by Software
25	String	
26	Write	
27	Timeout	
28	Store/System Error Timed	
29 - 31	Photo Dump Successful	

VIRTUAL STORE INTERRUPT PARAMETER

BITS

0-24	most significant 25 bits of VA
25-26	0X write access attempted 10 read access attempted 11 execute access attempted
27	if=1,indicates that a PE would occur on this access
28	if=1,indicates that the VSI occurred during interrupt sequence
29-31	=0,segment number greater than limit =1,segment not availeable(A=0) =2,segment number exceeded(non-paged segments) =3,page number greater than limit =4,page not available(A=0)

SYSTEM ERROR PARAMETER

Bits	
0-2	Not used
3-4	00 S/W SE 01 Irrecoverable failure H/W 10 Recoverable ISS error 11 Irrecoverable ISS error
5-7	Not used
8-11	ACR
12	Not used
13	Photo required- O=Yes,1=No
14	SSN+1 Dump- O=Yes,1=No
15	Photo type- O=Mini,1=Full

Software

16	Illegal VSI
17	Masked VSI
18	Masked PEI
19	Masked System Call
20	Masked OUT
21	Masked extracode
22	SSN is odd number
23	Segment Table format error
24	Entry forced by software
25	ACS=0 on initial activate
26-27	Not used
28-31	=F when dump forced

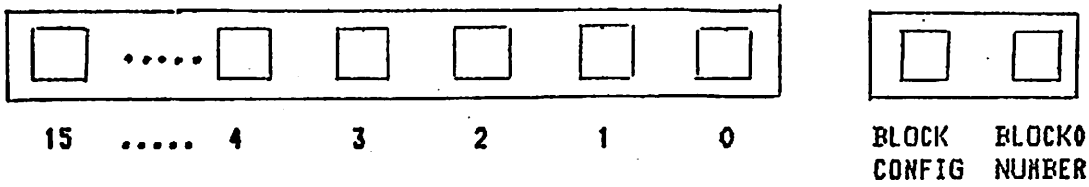
Hardware

16	Store fail
17	SCU detected error
18	Engine error
19	Scheduler error
20	SAU error
21	Engine timeout
22	SAU timeout
23	Comms fail
24	Not used
25	Hamming Log available
26-31	ISS module number

No: 6
Date: 11/7/78

2970: How to Reconfigure the Store on a SMAC

Use the bottom row of (mostly) blue keys in the store cabinet:



The BLOCK0 NUMBER key is grey, all other keys on this row are blue. Each 128 Kb block required to be on-line should have its corresponding key (0-15) pressed down. Effect the reconfiguration by pressing (and releasing) the adjacent BLOCK CONFIG key. The BLOCK0 NUMBER refers to the lowest numbered on-line block (generally real block 0). If the reconfiguration changes the lowest numbered block then the new BLOCK0 number should be set on the four grey keys on the row above the block configuration row and the BLOCK0 NUMBER key pressed and released. The fifth row of lights on the Store cabinet displays the current block configuration and BLOCK0 number.

N.B. The Store need not be contiguous except that the EMAS Supervisor loader requires blocks 0 and 1 of SMAC 0 to be configured on-line.

John Maddock

No: 7
Date: 9/10/78

2950: Notes on Transfer Control Blocks (TCBs) for People Who Are Used to the GPC

This Note is intended as a commentary on pages 30-40 of the ICL Product Specification document PSD 2.5.3, an extract of which is available on request. The diagram is taken from the document.

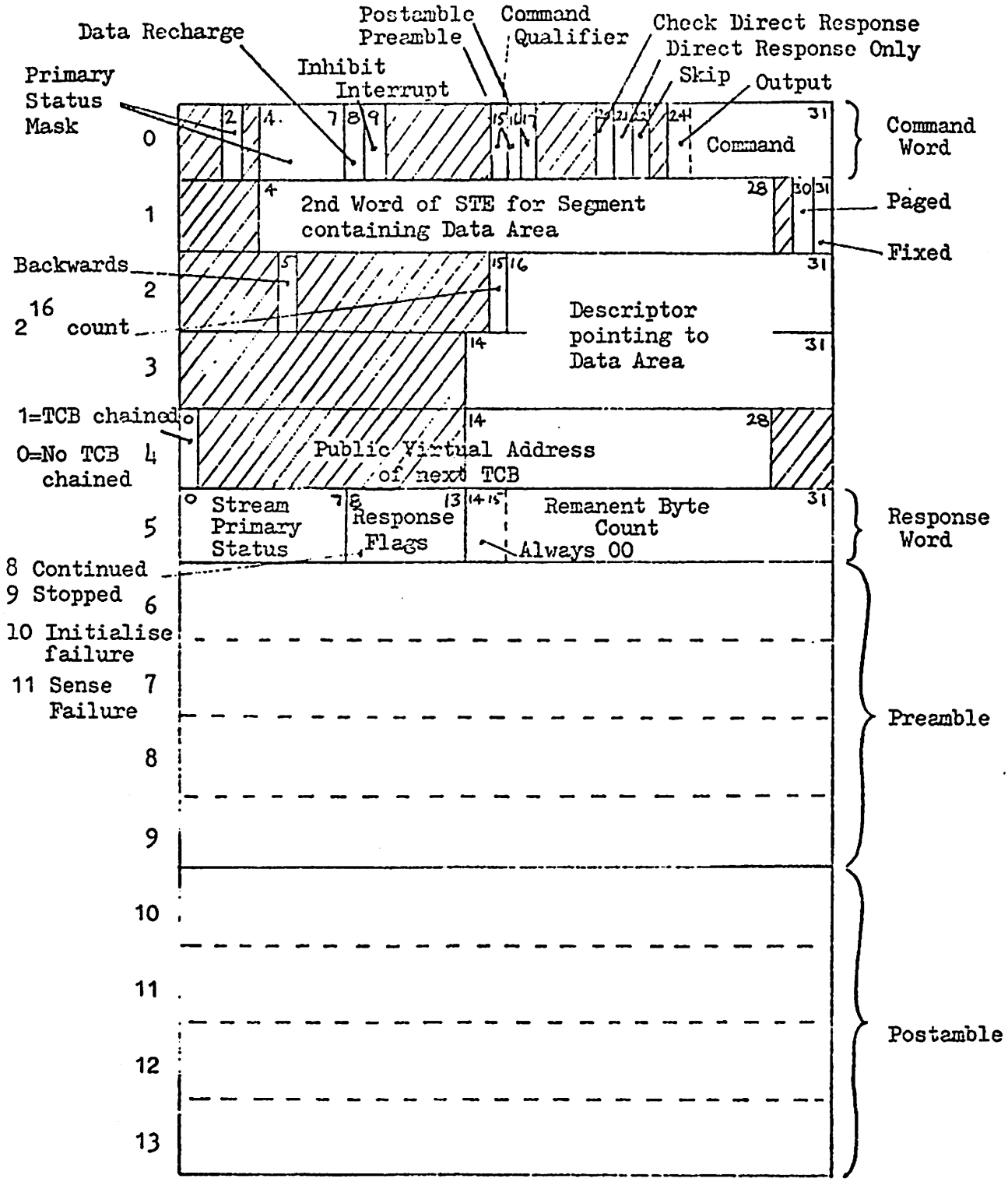
TCB Word 0

bits 0,1	not used
bit 2	if set, suppresses termination on "attention"
bit 3	not used
bits 4-7	suppress long block, short block, condition X, condition Y ≡ bits 8 to 11 of logic block entry
bit 8	data recharge from next TCB: equivalent to bit 4 of LBE
bit 9	inhibit interrupt: converse of bit 3 (PCI) in LBE (could also serve for bit 2 of SAW flags in STREAM SAW in H/W comm. area)
bits 10-14	not used
bit 15	command qualifier: 0 for "device", 1 for "stream"; same as bit 15 in LBE (?)
bit 16	preamble valid: controls initialisation, like RCB word 0 bit 16
bit 17	postamble valid: controls collection of status after unsuccessful transfer
bits 18,19	not used
bits 20,21	only relevant with 1900 SI peripherals
bit 22	for read commands: don't write data to store (skip). ? Not same as LBE bit 2.
bit 23	not used
bit 24	output (this seems to be nonsense, except for stream commands)
bits 24-31	command (in actual fact bit 31 seems to signal 'output'). Whichever bit actually indicates 'output' should correspond to bit 0 of LBE

TCB Word 1

bits 0-3	not used
bits 4-28	copied from 2nd word of STE of data segment - real address of segment or page table
bit 29	not used
bit 30	paged (same as bit 1 in 1st word of STE)
bit 31	fixed - copied from 2nd word of STE; must be 1, unless you have a command to transfer no data - then it must be 0

This substitutes for the sort of information which is found in word 1 of RCB.



Notes

(i) Shaded areas are ignored by IO System but those in word 0 are reserved (for hardware use).

(ii) Word 0 must be on 2 word boundary in a Public Segment with the TCB in a contiguous area not crossing a page boundary.

The Transfer Control Block.

TCB Words 2 and 3

Data descriptor. Must be a byte or string descriptor. Only the bound and address-within-segment are significant, and only 16 bits of the bound are used - i.e., byte count in word 2 bits 16-31, and address in word 3 bits 14 to 31. A zero byte count is taken to mean 2**16. S/w should have word 2 bit 15 =1 in that case, and =0 in all others. Word 2 bit 5 must be set for "backwards" transfers (? corresponds to bit 1 of LBE).

These two words correspond to the address list entry.

TCB Word 4

Public virtual address of next TCB - only bits 0 and 14-28 are significant. Bit 0 must be 1 (public address). If it is zero (local address) the chain is terminated. Therefore this corresponds to bit 4 V bit 5 in LBE (data chain V command chain). Only address-within-segment (bits 14-28) is significant because new TCB must be double word aligned and in same segment as previous TCB.

TCB Word 5

bits 0-7	stream primary status ≡ bits 16-23 of word 0 of STREAM RESPONSE in Controller H/W Comm. Area
bits 8-13	response flags bit 8 stopped bit 9 continued bit 10 initialise failure bit 11 sense failure
bits 16-31	remanent byte count ≡ bits 14-31 of word 1 of STREAM RESPONSE

TCB Words 6-9

Preamble area

word 6	initialise data (among other things) ≡ word 6 of RCB

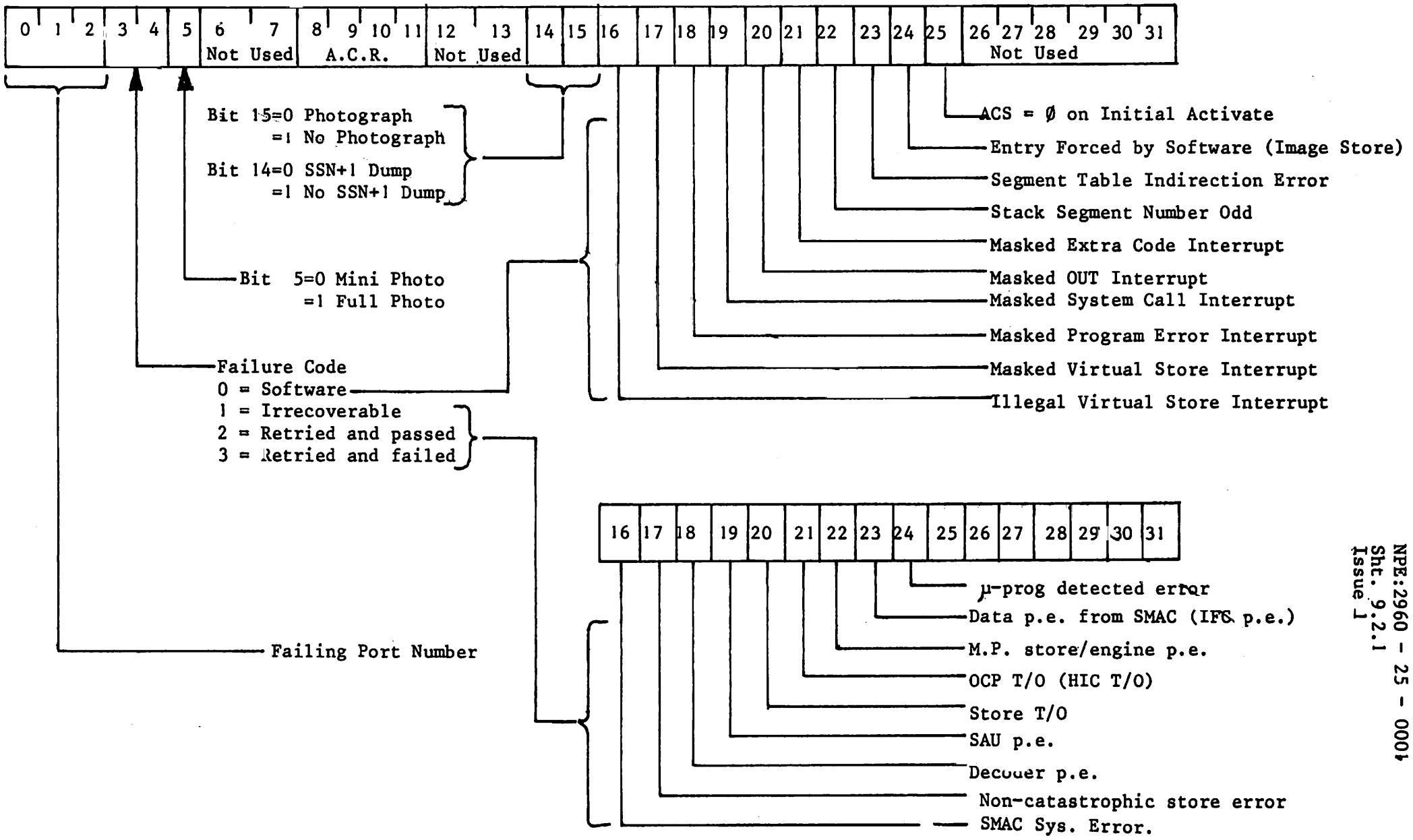
TCB Words 10-13

Postamble area. Gets sense bytes - secondary and tertiary status bytes.

J. Wexler

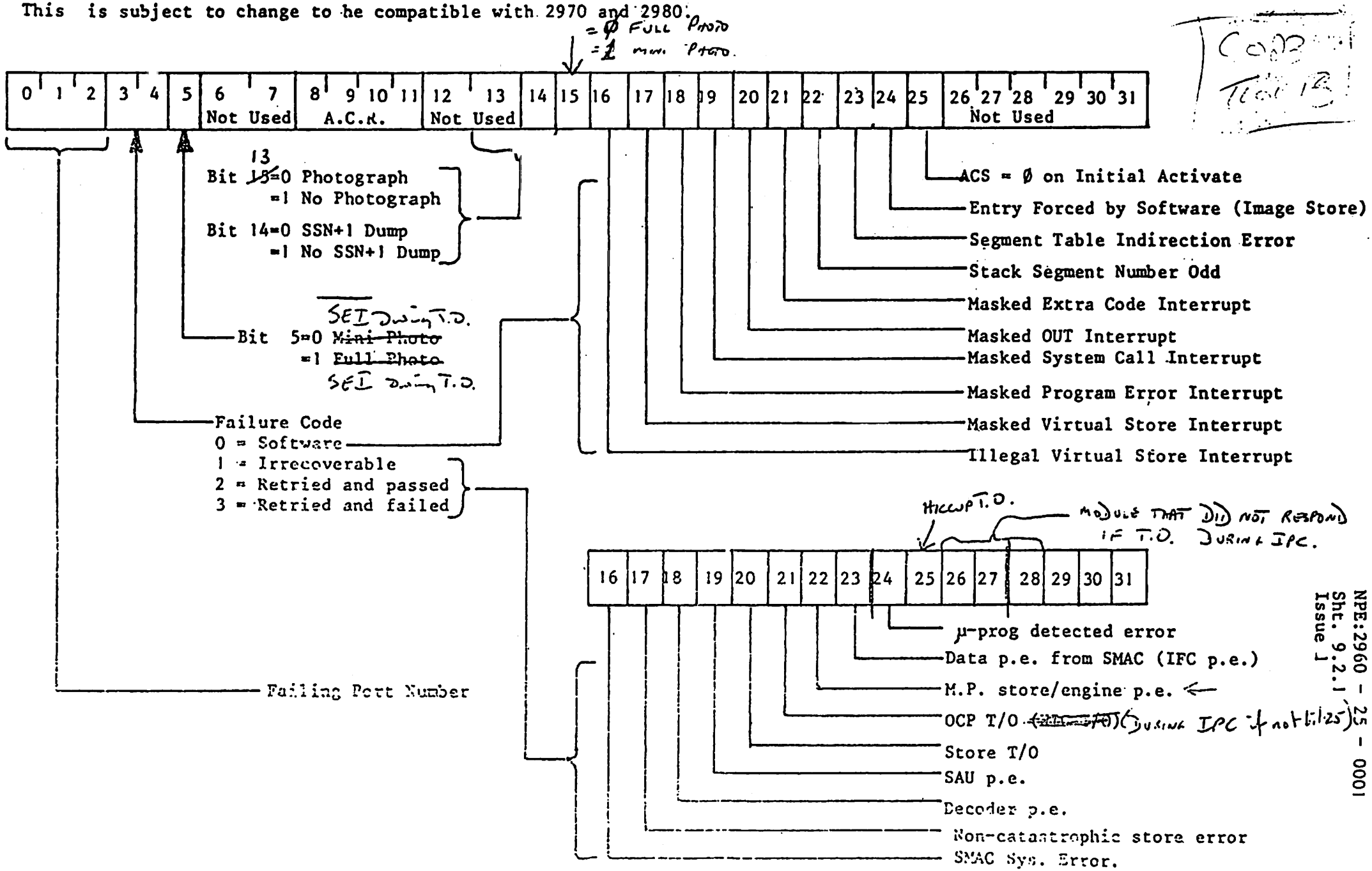
2960 5TEM ERROR INTERRUPT PARAMETER

This is subiect to change to be compatible with 2970 and 2980.



2960 SYSTEM ERROR INTERRUPT PARAMETER

This is subject to change to he compatible with 2970 and 2980.



NPE:2960 - 25 - 0001
Sht. 9.2.1
Issue 1

FIXED LOCATIONS IN WORKING STORE (N.R.)

a) Range Defined Registers Addr. by A/- or I/-

Words Ø.1	ACC Ø
2.3	ACC 1
4.5	ACC 2
6.7	ACC 3
8.9	DR Ø
10.11	DR 1
12.13	SF
14.15	B

b) Registers Addr. by C/- or C/B (For these C = # 10)

Addr. 16 (#1Ø)	Reg. SSN/LNB
17 (#11)	Prog Status Regs PSR (also STACK #1E)
18 (#12)	XNB
19 (#13)	SSR
20 (#14)	SSN
21 (#15)	LTB
22 (#16)	MBI (DETAIL SEE 3.7.3)
23 (#17)	RTCX
24 (#18)	IRS (Int Reg)(DETAIL 3.7.4)
25 (#19)	PAR(Parameter)
26 (#1A)	LSTBØ
27 (#1B)	LSTB1
28 (#1C)	IM1 (Int. Mask)
29 (#1D)	IM2 (Commissioning INT. Mask)
30 (#1E)	PSTBØ
31 (#1F)	PSTB1
32 (#20)	
33 (#21)	
34 (#22)	SI PAR
35 (#23)	SIV
36 (#24)	DPC
37 (#25)	RPAR

Block 1

FIXED LOCATIONS IN WORKING STORE (CONT.)

Addr.		Reg.	
	38 (##26)	HANDKEYS	
	39 (##27)	ERROR HANDLING OPTns. (DETAIL SECT.9)	
	40 (##28)	UPPER LIGHTS	
	41 (##29)	LOWER LIGHTS	
	42 (##2A)		
	43 (##2B)	IPL DATA	
	44 (##2C)	OPER LIGHTS SLAVE	
	45 (##2D)	PEI PARAM	
	46 (##2E)	W.S. IDENT.	
	47 (##2F)	SEI PARAM.	
	64((#40) - 71(=47)	1900 ACC's 0-7	1900 ONLY
	72 (#48)	FPACØ	1900 ONLY
	73 (#49)	FPAC1	1900 ONLY
	74 (#4A)	DATUM	1900 ONLY
	75 (#4B)	LIMIT	1900 ONLY
	76 (#4C)	SMO	1900 ONLY
	77 (#4D)	ESR	1900 ONLY
	78 (#4E)	MEB	1900 ONLY
	79 (#4F)	OBEY	1900 ONLY
	82 (#52)	TGT BLK ADD	
	94 (#5E)	LAST 7 BITS of PC,LOG POINTER	
	96 (#60-DF)	PC LOG AREA	

OPERATING INSTRUCTIONS FOR RUNNING T.C.S.S. UNDER EMAS.

HAVING LOGGED IN TO E.M.A.S. AS NORMAL PROCEED AS FOLLOWS :-

COMMAND: LTP XXXX WHERE XXXX IS THE 4 CHAR PROG. NAME.
(FOR THE FOLLOWING EXAMPLE WE SHALL ASSUME XXXX = EDA

TCSS 5.2

PLEASE SELECT UNIT FOR DEV0: YYYY WHERE YYYY = ALPHA NUMERIC MNEMONIC FOR 1ST DEVICE TO BE TESTED
SEE NOTE 1
AND NOTE 3

YYYY USED AS DEV0

PLEASE SELECT UNIT FOR DEV1: ZZZZ WHERE ZZZZ = ALPH NUMERIC MNEMONIC FOR 2ND DEVICE TO BE TESTED
SEE NOTE 1
AND NOTE 3

ZZZZ USED AS DEV1

PLEASE SELECT UNIT FOR DEV2: FORGO i.e. NO FURTHER TEST DEVICES

PROGRAM EDAR(1) LOADED SUCCESSFULLY

>- TCSS/TEST PROGRAM HAVE NOW BEEN ENTERED AND ALL STANDARD TOSD/TCSS COMMANDS, WITH THE EXCEPTION OF BREAKIN, ARE VALID. FOR BREAKIN SEE NOTE 2.
FOR LIST OF VALID COMMANDS SEE TCSS TPUM

>- LGT CAUSES TCSS/TEST PROGRAM TO BE DELETED AND PROCESS TO BE LOGGED OUT

NOTES ON RUNNING T.C.S.S. UNDER E.M.A.S.

1/ DEVICE MNEMONICS

AS T.C.SS / TEST PROGRAMS REQUIRE 4 CHAR MNEMONICS (NORMALLY TWO ALPHA FOLLOWED BY TWO NUMERIC CHARS) AND SOME EMAS MNEMONICS ARE ONLY THREE CHARS LONG CERTAIN GROUND RULES MUST BE OBSERVED. THESE RULES ARE AS FOLLOWS :-

i/ EDS MNEMONICS — USE EMAS MNEMONIC

ii/ M/T MNEMONICS — IF THE LEAST SIGNICANT DECK IS M10 THEN THE CONCEPTUAL EMAS CLUSTER MNEMONIC BECOMES MC10 FOR M20 THEN MC20 ETC.

iii/ SLOW DEVICE MNEMONICS — WHERE THERE ARE ONLY 3 CHARS IN THE EMAS MNEMONIC THEN A LEADING Ø IS PLACED BEFORE THE NUMERIC CHAR i.e.

CRØ BECOMES CRØØ
LP1 BECOMES LPØ1
LP11 REMAINS LP11

2/ BREAKIN

AS "CONTROL ∝ A" IS MEANINGLESS TO EMAS BREAKIN IS ACHIEVED BY HITTING "ESCAPE" ("CONTROL, SHIFT ∝ K" ON LIMITED SET T/T COMPATIBLE VDU'S) AND WHEN PROMPTED INT: REPLYING "BI". MULTIPLE BREAKINS ARE PERMISSABLE AND WILL CAUSE INSTANT BREAKIN TO TCSS WHEN THE TEST PROGRAM IS RUNNING.

3/ AVAILABILITY OF DEVICES FOR TESTING

BEFORE DEVICES CAN BE ACQUIRED BY TCSS FOR TESTING THEY MUST BE IN THE CORRECT SYSTEM STATE. THIS CAN BE ACHIEVED AS FOLLOWS :-

i/ E.D.S

IT IS SUFFICIENT TO MOUNT A NON-STANDARD DISC (i.e. EDAR DISC) ON THE DRIVE(S) TO BE TESTED.

ii/ M/T's

FOR M/T'S IT IS NECCESSARY TO TYPE THE FOLLOWING AT THE MAIN CONSOL

V/ MXX MODE = OFF

WHERE MXX IS [redacted] THE EMAS MNEMONIC FOR THE DEVICE. n.b. THIS MUST BE DONE FOR EVERY DECK IN THE CLUSTER TO BE TESTED.

iii/ SLOW DEVICES

FOR SLOW DEVICES IT IS NECCESSARY TO TYPE THE FOLLOWING AT THE MAIN CONSUL

S/ XXX LIMIT Ø

S/ ABORT XXX WHERE XXX IS THE EMAS MNEMONIC OF DEVICE TO BE TESTED.

SHOULD THE DEVICE(S) NOT BE IN THE REQUIRED STATE WHEN TCSS ATTEMPTS TO ACQUIRE IT THEN THE MAIN OPERATOR CONSUL WILL BE PROMPTED AS FOLLOWS :-

"PLEASE CONFIGURE OUT XXXX"

TWO COURSES OF ACTION MAY THEN BE TAKEN

i/ THE DEVICE CAN BE MADE AVAILABLE AND THE PROMPT ANSWERED "PDN" (PROMPT DONE), T.C.S.S WILL THE CARRY ON AS NORMAL.

ii/ THE PROMPT CAN BE ANSWERED "CD" (CANT DO). AN APPROPRIATE MESSAGE WILL BE OUTPUT AT THE TERMINAL AND TCSS & TEST PROC. WILL BE DELETED AND THE SESSION TERMINATED.

n.b. AS EMAS IS NOT BLOCK STRUCTURED RESOURCES ARE NOT

NECCESSARILY SURRENDERED AT THE END OF SESSION. THE TCSS-EMAS SUPPORT PACKAGE WILL, OF COURSE, RELINQUISH ALL DEVICES ALLOCATED TO IT AT THE END OF SESSION. THIS, HOWEVER, PRE-SUPPOSES A CLEAN EXIT FROM TCSS. A SITUATION CAN ARISE WHERE TCSS IS NOT EXITED FROM IN A TIDY MANNER e.g. IN THE EVENT OF A CRASH IN THE ENGINR. PROCESS DURING RUNNING OF TCSS/TEST PROGRAM. THIS CAN LEAD TO A SITUATION WHERE ENGINR "OWNS" A DEVICE(S) BUT T.C.SS/TEST PROGRAM CANNOT USE IT AS ANY ATTEMPT TO ALLOCATE IT FAILS AS IT IS ALREADY ALLOCATED. SHOULD TCSS/TEST PROGRAM CRASH DURING RUNNING IT IS ADVISED THAT ANY ALLOCATED DEVICES ARE MANUALLY RELEASED. TO THIS END A SPECIAL COMMAND "RELEASE" HAS BEEN PROVIDED. THIS TAKES THE FORM

COMMAND: RELEASE XXXX where XXXX IS THE EMAS MNEMONIC

IT IS FURTHER ADVISED THAT IN THE EVENT OF A CRASH NO ATTEMPT IS MADE TO RE-ENTER TCSS WITHOUT FIRST LOGING OUT AND LOGING IN AGAIN. IT WON'T WORK AS CERTAIN TCSS VARIABLES WILL NOT HAVE BEEN RE-INITIALISED.

GENERAL DESCRIPTION SRU

PART 'D' SECTION 1

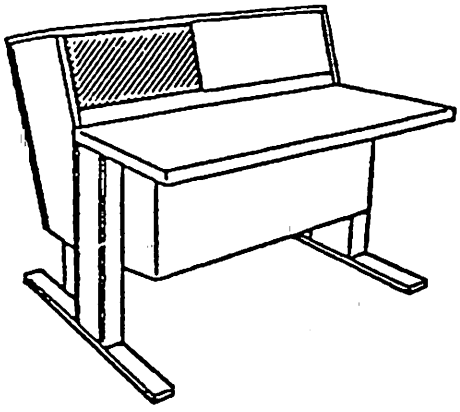
1.1 Major system units SMAC, SAC and OCP on a dual system are configured by the (SYSTEM RECONFIGURATION UNIT) SRU. System units SAC and OCP are visible to the system via the ports of SMAC. The SRU uses HI HO control over transmitters and receivers of the port interfaces, and also to isolate the (DIRECT WIRED INTERFACE) DWI. The DWI is an interface between SAC and OCP, OCP and OCP, in a dual configuration.

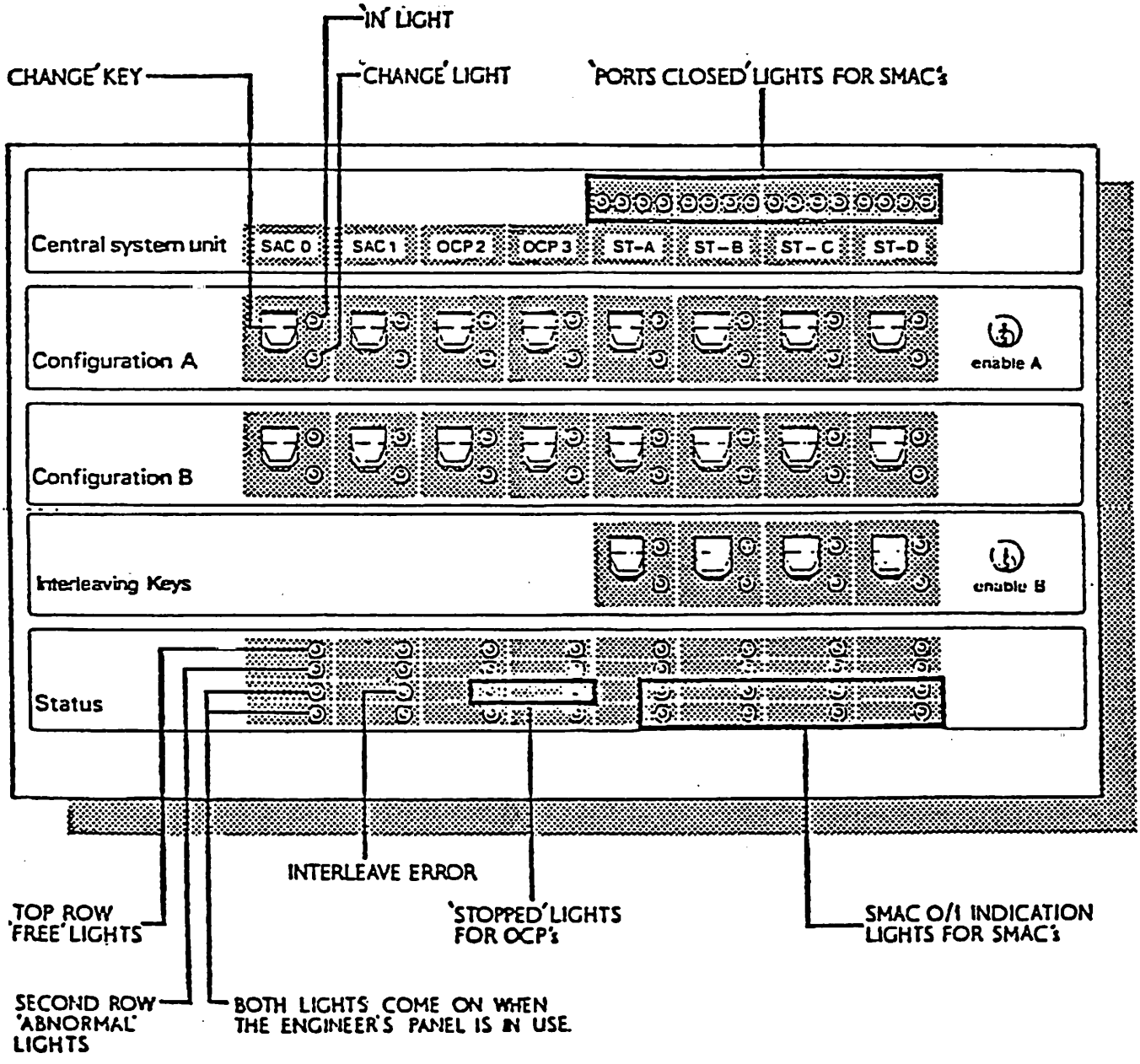
The configuration is set on the SRU at load time. Thereafter, the configuration is set by writing to an image store in SMAC to enable dynamic reconfiguration. This software can be envoked by the command (CONFIGURE DEVICE STATUS) CDS or by a software decision to remove a unit from the configuration if not being used.

1.2 The SRU

This unit has two main panels accessible to the operator. Both the panels are divided into two. The left hand panel is the reconfiguration panel. This panel has sixteen switches, eight in either half. The 2976 has an extra 4 switches, set below the standard sixteen switches. These are for interleaving configuration.
A dual configuration can be set in either half 'A' or 'B'. If a configuration is set in both 'A' and 'B', the system is said to be partitioned. In a dual configuration all units are set in one half. To set a configuration, all switches must be put down to the change position, this will illuminate the red change light below the switch. A change can only take place if the unit is in the free state, this is indicated by a green light at the bottom of the panel. A unit is set into a configuration by the operation of the appropriate enable key 'A' or 'B'. All keys selected to the change state must be put to the normal position (up), because if the enable key for that configuration is pressed again with the key in the change position, it will be removed from the system and become free.

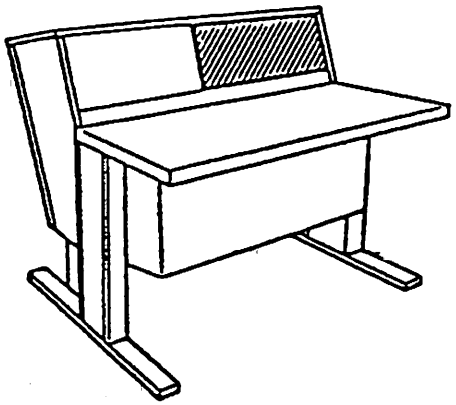
Other indicators on this panel are port closed lights which are positioned above the SMAC switches. These indicate when SMAC ports are closed. Along the bottom of the panel there are four rows of indicator lights. The top row of these indicate when the unit directly above it is in a 'free' state i.e. not in either configuration. The second row indicate an abnormal condition on the unit, this can be a power supply fail or an engineers switch left in an operative state. The third and fourth rows are not symmetrical and therefore will be described individually. The two lights at the left hand end indicate when lit that the engineers panel is connected to the SRU.

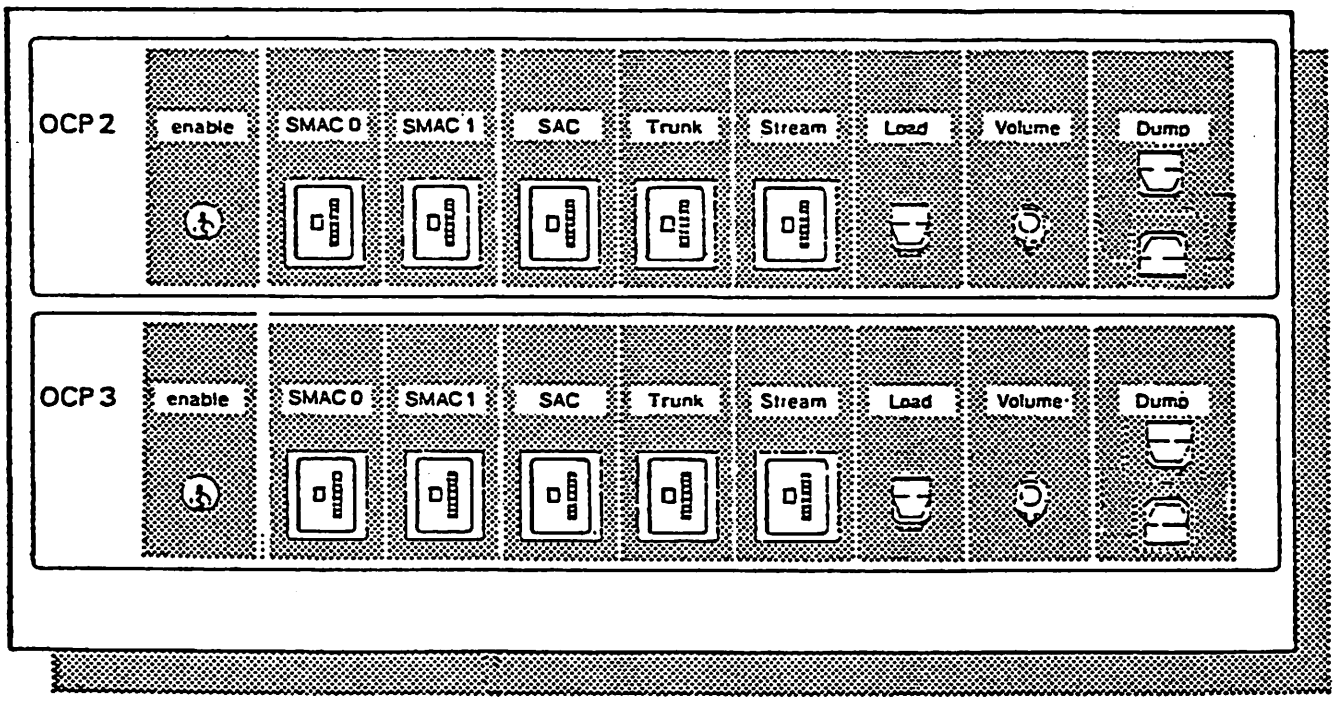




2976 RECONFIGURATION PANEL

© International Computers Limited





2976/80 BOOTSTRAP PANEL

© International Computers Limited

The operation of the engineers panel will be dealt with in a separate section. The next indicator on the third row is only fitted to 2976 SRUs and indicates an interleaving error. This condition will occur when the interleaving rules are broken. The next set of lights are in the third row below the OCP change switches and indicate when the OCP is in the stopped state. The last set of lights are a group of four and relate to the SMAC number. The 2976 differs in this respect to all other 2900's the SMAC number here is always in pairs module 8 apart.

1.3 Bootstrap Panels

The bootstrap panels are situated on the right hand side of the SRU. These two panels are divided in two halves - one for OCP2, the other for OCP3. Each half contains thumb wheels to select the IPL path, a volume control and switches for load and dump. The panels vary in layout within the 2900 range. 2980 and 2976 have an extra thumb wheel switch, as these machines have the facility to interleave.

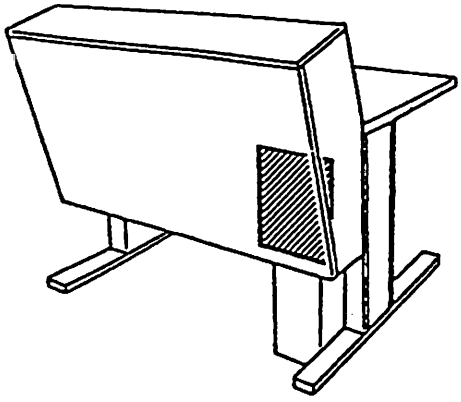
The operation of this panel is reasonably self-evident. The SMAC or SMAC's and SAC numbers are selected on the thumbwheels the SMAC number is in the range A to D. This letter relates to the ST letter selected on the reconfiguration panel. The letter selected will deem that SMAC to be SMAC Ø. The SAC number is in the range 0 to 1 as only two SAC are allowed on the system. The other two thumb wheels select the trunk and stream of the IPL path, both in the range 0 to F. When the selection is complete, the load or dump keys are operated simultaneously with the enable key. Note there are two dump keys for each panel which must be operated at the same time to envoke the dump sequence.

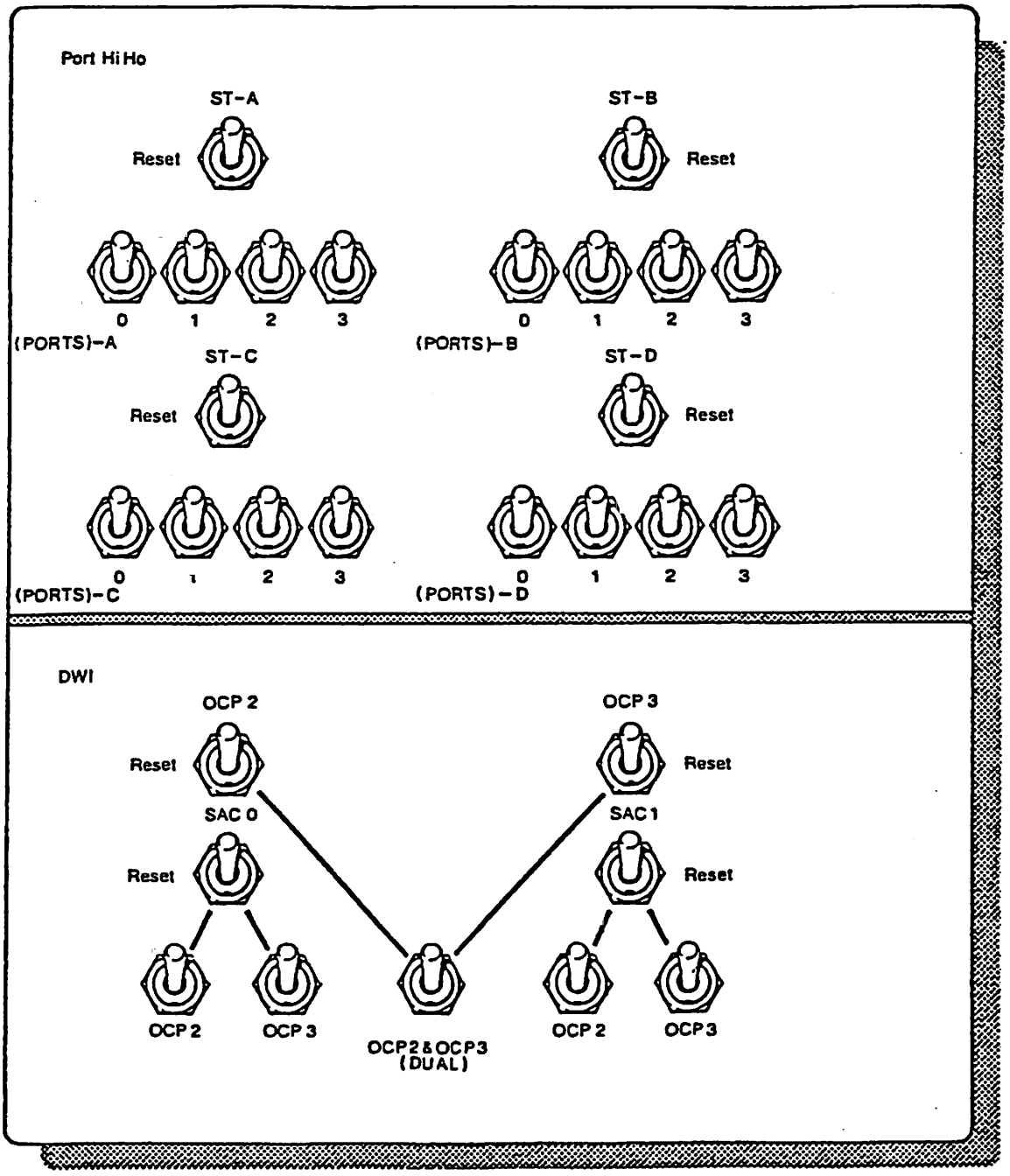
1.4 Engineers Panel

This panel is situated at the rear of the SRU behind the rear cover. Normally the connecting cables 1R and 2R will not be connected. As stated in section 1.2, when this connection is made a two-light display will illuminate on the operators reconfiguration panel. The purpose of the engineers switch panel is if the power supply fails in the SRU, a configuration can be set on the engineers switches to simulate the original configuration to enable the customer to continue.

1.5 Engineers Panel Operation

The switches on this panel are operative when in the (up) position. There are twenty one toggle switches and eight bias switches. The panel is divided into two halves, the upper half is port control, the lower is DWI control. To set a dual configuration using two SMACs, two SACs and two OCPs, put all toggle switches 0, 1, 2, 3 up on port 'A' and 'B'. Put the lower five toggle switches up. Operate the bias switches for port 'A' and 'B'. Operate the bias switches on the DWI half for OCP2, OCP3, SAC Ø and SAC1. The bias switches produce local IPL signals and open the path for the IPL. Complete system load as per section 2.4, 3.4 4.7 for 2960, 2970 and 2976 respectively.





ENGINEERS PANEL

NOTE: THE POSITION OF THE KEYS SHOWN IN THIS FIGURE IS THE 'OPERATIVE' POSITION

© International Computers Limited

To partition neatly in two balanced halves using one SMAC in each half, set the switches as follows :

Port 'A' switches - Put switches 0 and 2 to 'up' and switches 1 and 3 to 'down'.

Port 'B' switches - Put switches 1 and 3 to 'up' and switches 0 and 2 to 'down'.

DWI half left hand switches set, OCP2 switch 'up' OCP3 switch 'down'.
Right hand switches set OCP3 switch 'up' OCP2 switch 'down'.
Set centre toggle switch 'down' this will break the DWI interface between OCPs.
Operate bias switches for Port 'A', port 'B', OCP2, OCP3, SAC Ø and SAC 1. This action will produce local IPL and set the port paths for IPL.
Complete the system, load as per sections 2.4, 3.4 and 4.7 for both OCPs.

Alternative Partition

To set a partition using SMAC 'A' SAC1 and OCP2 in one half, and SMAC 'B' SAC Ø and OCP3 in the other half.

Port 'A' switches, set switches 1 & 2 to 'up', set switches Ø and 2 'down'.
Port 'B', set switches Ø and 3 to 'up', set switches 1 & 2 'down'.

DWI switches, left hand switches, set toggle switch OCP3 to 'up', set toggle, switch OCP2 to 'down'.

Right handswitches set toggle switch OCP2 to 'up', set toggle switch OCP3 to 'down'

Operate the bias switches for Port 'A', Port 'B', OCP2, OCP3, SACØ and SAC1, this action will produce local IPL and set the port paths for IPL.

Complete the system load as per sections 2.4, 3.4 and 4.7 for both OCPs.

Interleaving Switches (2976 only)

1.6 There are four interleave switches one for each store. To select stores for interleaving, the switches must be put down to the change position. Also, the associated SMAC switch must be put to the change state. Interleave switches and SMAC switches must be selected in pairs, the following are allowed :

A&B, A&D, C&D, B&C.

Any other combination of pairs will result in the 'error' light being illuminated. When the switches are selected to the change state, the enable key A or B to which they are to be configured is pressed. If the action is successful, the 'IN' light will be illuminated for each change.

S.R.U.

1.7. GENERAL DESCRIPTION APPENDIX FOR 2976 S.R.U.

BOOTSTRAP PANEL

On this panel there are two 'force SMAC' thumbwheels for each OCP row, namely SMAC 0 and SMAC 1.

The 'force SMAC 1' thumbwheel also has a blank position to accommodate the need for not having a SMAC 1. These thumbwheels are to conveniently facilitate a four SMAC interleaved system.

Certain numbering restrictions are imposed when interleaving in addition to the physical restrictions mentioned in section 1 - 6:- ST A and ST C are termed the even SMACs, ST B and ST D are termed the odd SMACs.

The convenient valid states are:-

SMACs Interleaved	SMAC NOs
A and B	0 and 8 or 1 and 9
C and D	0 and 8 or 1 and 9
B and C	8 and 0 or 9 and 1
D and A	8 and 0 or 9 and 1

It can be seen from the above table that:-

1) You cannot interleave 2 even SMACs
2) You cannot interleave 2 odd SMACs
3) The even SMAC must have the lowest number

On a duals site all of the above options cannot be used without altering the SMACs natural number.

For a four SMAC interleaved configuration it is usual for the SMACs to have the following addresses, but not obligitory.

ST		NATURAL No
SMAC A	=	A
SMAC B	=	8
SMAC C	=	B
SMAC D	=	9

All 4 SMACs interleaved Force 0 = A or C

Force 1 = C or A

It can be seen from this numbering system that if you partition, in two balanced halves, only one configuration can be interleaved, i.e. whichever configuration has SMAC 'D', with natural number of 9, cannot be interleaved, the reason being that you must have a SMAC 0.

On a 2976 SRU the SAC thumbwheel has a blank position. This thumbwheel is used to allow the SAC activate and the SAC interrupt in the IPL OCP. When doing an IPL in a dual configuration with two SACs and two OCPs, the non-IPL OCPs associated SAC thumbwheel must be set to blank. This puts the non-IPL OCP into a suspended state.

SRU CONFIGURATION TECHNICAL OPERATION

1.8. General

When the SRU is switched on, all logic is held in a 'frozen' state, a mechanical timer is situated at the rear of the unit. The timer is adjustable from 0 - 60 seconds, and should be set to 15 seconds.

A configuration is set by putting the appropriate change switches down. The action can only successfully be completed if the unit to be changed is in the 'FREE' state. The action of the change switch is to prime a binary, the setting of binary is clocked by clock pulse 2 when the reconfiguration enable key is operated.

The enable key produces two clock pulses, clock 1 and clock 2. Clock 1 is a 5ms pulse separated from clock 2 by 5ms. Clock 2 is a 30ms pulse. Clock 1 sends a local IPL signal to the SMAC, this will set an OCP port to 'OPEN'. The SAC ports are not established until clock 2 time, and are set in the SMAC at bootstrap load time.

Clock 2 sets the configuration selected into the logic, where it is locked into binaries and then via gating control operate logic relays. The relays control the HI HO lines of each units connected interface, this also includes the direct wired interface.

1.9. Engineers Switches

As can be seen by the accompanied logic drawings, the engineers switches sit across the relay outputs. Thus in the event of a power failure, the selected configuration can be simulated (see general SRU description). It is also evident from the drawings that the engineers switches must not be left in an operative position, as they override the relays if the engineers panel connecting plugs are left unconnected, this condition will be indicated by two leds on the reconfiguration panel.

1.10. Interleaving Configuration

The logic employed is designed to make certain the rules for interleaving are obeyed (see Table 1). If the rules are broken, then an error signal will be produced, and an indication of this shown on the operators reconfiguration panel. If allowed pairs are selected, this will operate the appropriate relays to signal to the OCP and SAC, to treat that pair as interleaved store.

TABLE 1

	STA	STB	STC	STD
allowed interleaved pairs	✓	✓		
	✓			✓
			✓	✓
		✓	✓	

PART D SECTION 2

DUALS COMMISSIONING

2.1.0 This document assumes that the system reconfiguration unit (SRU) has been physically installed.

2.2.0 Relevent Documentation

SRU Technical Description 65/94682
SRU Installation Description 65/94690
TPUD Part B Volume 1 (Test Program IPCT, Dual)
TPUD Part B Volume 3 (Test Program NF 2960)

2.3.0 SRU Static Checks

3.1 Check the following areas and observe that no damage has occurred during transit.
3.1.1 All covers fit and operate correctly and are not damaged
3.1.2 That all connectors, plugs, sockets, etc. are fitted and located correctly.
3.1.3 All package pieces have been removed as per installation data Document No. 65/94690.

2.4.0 Power Supplies

4.1 Warning hazardous energy.
4.1.1 Check mains switch is off
4.1.2 Carry out check for short circuit on LAMDA + 5 volt power supply.
4.1.3 Using a DVM set to OHMS scale, with 0.5 range selected measure resistance across TBI and TB2 located behind lower front cover reading should be approximately : 5 OHMS

2.5.0 Switch on Mains switch located at rear of unit.

5.1 Using a DVM set volt range check between TBI and TB2 for + 5 volt ± %5 if this is not present adjust power supply by means of potentiometer located at the right side of the LAMDA power supply.

2.6.0 Configuration Mode checks

6.1 With power applied to the unit and all switches in the up position, check that all the 8 (Free) lights are illuminated.
6.1.1 Switch all 16 selection switches down and press (A) enable key once. Observe that all 8 (IN) lights are illuminated. Press enable (B) key once and observe there is no change in the configuration.

2.6. continued

6.1.2. Press enable A key once and observe all 8 (Free) lights are illuminated and all 8 configuration (A) (IN) lights are extinguished. Press enable (B) key once and observe all 8 (Free) lights are extinguished and all 8 configuration (B) (IN) lights are illuminated. Press enable (A) key and observe there is no change in the configuration. Press enable (B) key once and observe that all 8 configuration (B) (IN) lights are extinguished.

2.7.0. PARTITIONED SYSTEMS

Set switches on SRU reconfiguration panel for system (A) as in table (1) column (A). Press enable key once. Set switches on SRU reconfiguration panel for system (B) as in table (1) column (B). Press enable key once. Set OCP 2 section of bootstrap panel to Port Trunk and Stream of the media selected for system load. Press the associated load switch and enable key on bootstrap panel simultaneously. Repeat the above procedure for system (B) and OCP 3 section of bootstrap panel.

2.7.1. Repeat 7.0. procedure for columns (C) and (D) table(1).

2.8.0. Set switches on SRU for system as in table (1) column (E). Press enable key once. Set OCP 2 section of the bootstrap panel to Port Trunk and Stream of media selected for system load.

2.8.1. Load and run free standing test "IPTC" TPUD Part A Volume (1) when complete, load and run free standing test 'Dual'.

2.8.2. Repeat the above procedure using OCP 3 section of bootstrap panel using table (1) column (F).

TABLE 1	A	B	C	D	E	F	G
SAC 0	IN			IN	IN	IN	IN
SAC 1		IN	IN		IN	IN	IN
OCP 2	IN			IN	IN	IN	IN
OCP 3		IN	IN		IN	IN	IN
ST -A	IN			IN	IN	IN	IN
ST -B		IN	IN		IN	IN	IN
ST -C	(IN)			(IN)	(IN)	(IN)	(IN)
ST -D		(IN)	(IN)				(IN)

ST -C fitted only to 2970 and 2976/80 systems

ST -D fitted only to 2976/80 systems

2.9. DUALS COMMISSIONING S.R.U. APPENDIX 2976

Set switches on SRU for system (A) as in table (1) column (G) with the SMACs non-interleaved. Press enable key once. Ensure the SMACs are not forced to 0 or 1 using the SMAC engineers panel. Set, on OCP 2 section of bootstrap panel, SMAC 0 thumbwheel to A and SMAC 1 thumbwheel to B. Press associated enable key and load switch on bootstrap panel simultaneously. Check that the SMAC 0/1 indicators on the reconfiguration panel are lit accordingly. Repeat the above procedure checking that each SMAC can be forced to 0 and 1. Repeat the procedure for OCP 3 section of bootstrap panel.

N.B. A reset from the OCP with the undump key in the down position will cause the SMACs to revert to their natural numbers.

2.9.1. Check that any illegal interleave conditions (see general description section 1.6.) lights the "Interleave Error" L.E.D. on the reconfiguration panel.

2.9.2. Set up switches on SRU for system (A) as in table (1) column (G) with all four SMACs interleaved (see SRU general description section 1.6.). Press enable key once. Assuming SMAC natural numbers are set to:-

SMAC A = A

SMAC B = 8

SMAC C = B

SMAC D = 9 then set, on OCP 2 section of bootstrap panel SMAC 0 thumbwheel to A and SMAC 1 thumbwheel to C. Press enable key and load switch simultaneously. Using OCP engineers panel, check that the stores can be written to and read from, and that the 'interleaved STATS' are set (STIF Frame 6 Line 9 bits 32-35). There is no visible indication of interleaving in the SACs, therefore, using a scope, check the interleaving signals in both SACs. (SAC 3A/3B ED's Page 14.7 Pins AM 2L, AM 1R, AM 3L and AM 3R).

2.9.3. Set switches on SRU for system (A) as in table (1) column (G). Press enable key once. Set OCP 2 section of bootstrap panel to Port, Trunk and Stream of media selected for system load. Set SAC thumbwheel on OCP 3 section (non-IPL OCP) to blank. Press enable key and load switch simultaneously. Check that the non-IPL OCP (i.e. OCP 3) goes to a suspended state, i.e. COM 8 (frame 10 Line 8 bit 10). Repeat the above procedure using OCP 3 section for the selected media address and setting SAC thumbwheel to blank on OCP 2 section.

2.10. ENGINEERS SWITCHES SRU

2.10.1. To Partition System

Port H1 HO Switches

Switch ST-A switch 0 & 2 up. Operate ST-A reset switch

Switch ST-B switch 1 & 3 up. Operate ST-B reset switch

Left-Hand Switches (LOWER)

Switch OCP2 switch up switch OCP3 switch down, operate SACØ reset switch

Right-Hand Switches (LOWER)

Switch OCP3 switch up switch OCP2 switch down, operate SAC1 reset switch

Carry out IPL sequence from OCP engineers panel (2970) ECP (2960)

When load sequence is complete, check that selected ports are indicated on the SRU front panel leds.

2.10.2. To Dual System

Port H1 HO switches

Set all port switches of operational SMACs to up, activate all operational ST reset switches.

DWI switches

Set lower 5 switches to up, activate all reset switches OCP2, SACØ OCP3, SAC1.

Carry out IPL sequence from OCP engineers panel (2970) ECP (2960)

When load sequence is complete, check that selected ports are indicated on the SRU front panel leds.

- SMAC Ports and numbers must be forced at the SMAC's and the OCP Engineers facilities used to 'IPL'.

Section 2.2. Creating a u code dump tape

	STEP	
ECP	1.	Type FI UTIL SICL 'SEND' and await flashing cursor
	2.	Unload TLOE cassette by depressing → and CNTRL keys together and replace with u code cassette.
	3.	Type SICL and 'SEND' and await completion of u code load to main store.
	4.	Type WWS 2B 2000 0000.
	5.	Unload u code cassette using → and CNTRL keys and replace with TLOE cassette.
M.T.	6.	Load a working tape on to lowest order deck on M/T cluster.
ECP	7.	Type DU pts 'SEND'



Port trunk and Stream of M/T unit

u code will now be dumped to working tape.

8. Repeat action 7 several times to reserve further copies of u code on the tape.

A. TCSS FACILITIES

See Appendix A for list of available test programs.

TCSS COMMANDS

LOAD_TEST_PROGRAM (LTP)	This macro establishes the environments for engineers test programs

PARAMETERS

MODULE = NAME OF TEST PROGRAM FILE
ASSIGNMENT = DEVICE/LINE/CONTROLLER MNEMONIC
ACTUAL UNITS = DEVICE MNEMONIC
EXTEND = LIBRARY NAME
ROUTE = SPOOLED/ONLINE
RESPONSE = RESULT)

ABANDON(ABA)	Abandons test program selection at earliest opportunity
ABORT(ABO)	Abandons current selection immediately
ATERROR_DEFAULT(ATEDEF)	Returns default option to calls from test program
ATERROR_RECOVER(ATEREC)	Returns recover option to calls from test program
ATERROR_ASK(ATEASK)	Removes the other two aterror conditions
CYCLE(CYC)	Causes a selection to be cycled
CYCLE_OFF	Cancels the cycle command
GO	Starts off test program selection
HOLD(HOL)	Causes a repeat option to be returned to test program
HOLD_OFF(HOLOFF)	Cancels the hold action
LOCK(LOC)	Causes repeat options to be returned to test program until a different h/w error occurs, the h/w operation is successful or the program is unable to repeat the operation.
LOCK_OFF	Cancels the lock action

ON_TP	Sets bits in test options word Eg, ON_TP ("3 5 9")
OFF_TP	Resets bits in test options word
RECOVER(REC)	When RECOVER option offered causes test program to perform defined recovery action
REPORT(REP)	When REPEAT option is offered causes the test program to repeat the failing operation
ROUTE(ROU)	Causes messages of the specified class of set of classes to be routed as instructed. Eg, ROU("20 BULK")
SELECT(SEL)	Causes an element to be selected. Activated by GO command. EG, SEL(ALL)
SET	Enables the setting of test parameters declared by the test program. Eg, SET UTIL = "E"
SELECTANDGO(SGO)	Equivalent to SELECT command followed by GO, eg SGO(ALL)
STEP(STE)	Executes the current selection one element at a time and halts at the end of each element
STEP_OFF	Cancels the STEP action
SUSPEND(SUS)	Suspends the test program until a GO command is issued
TERMINATE(TER)	Causes termination of the current element at the earliest possible opportunity but allows the rest of the selection to continue
WHATIS(WHA)	Puts on the oper details of the following: PARAMETERS ELEMENTS PARAMETER NAME TESTOPT

EXAMPLES OF RUNNING TCSS

TCSS tests with standard set of node calls can be run from oper by typing:

ET(TESTNAME)

The test units, which must be configured out, will be prompted for. Alternatively for a wider selection of required nodes the test can be input as a job as shown below:-

```
CUJ(:ENGINEER)
J(EDAR)
BEGIN
LTP(EDAR)              @will prompt for test unit, which must be
SET UTIL = "E"          configured OUT.@
SGO 99.8
SET UTIL = "N"
SGO 3
ABA
END
ENDJOB
****
```

Can also be run in MAC environment

The engineers OCF (S ENG or S 12) can be displayed for information on progress of a test.

4. DUAL SYSTEM ERRORS

4.1 Failing OCP

(a) 2960/2970

For any system error other than a timeout resulting from an access to the other OCP, the failing OCP will broadcast the fact that it has had a system error, and then 'go to sleep'. In the case of a timeout, the system error is not broadcast, as the remote OCP is presumed to be dead.

'Sleep' involves looping in the microprogram at the following addresses:-

2970	19I5	771 - 772
	7000	77B - 77C
	1000	783 - 784
2960	C14	1037 - 103E
	C17	10CC - 10D3

The microprogram will remain in this loop until a Block 2 Subblock 60/63 image store access is performed from the other OCP with Bit 12 = 0, i.e. do not suspend on completion. On 2970, the EKS key may be used to get out of this loop manually. On 2960, typing HA will take it to 'Tidystop'.

(b) 2976 Types

The following types of system error are broadcast:-

Irrecoverable H/W SEs, including failing retry.
Masked S/W SEs, i.e. SEI is masked in SSR.
This is only when SEI Analyser is runni_.

SGSE (FSEI)

For other crash conditions, e.g. MVSI or MPE, SEI-Analyser is entered in the detecting OCP, and will suspend the other OCP by accessing IS 42PØØØØ4. Note that this does not cause an SSN+1 dump, so PC, LNB etc, can only be obtained from the lights. The 'suspend' condition is shown by COMM8=1 (Frame 10, line 8, bit 10).

On receipt of a broadcast SEI, the H/W 'Masks' MULT until the SEI parameter has been read from the failing OCP, at which point it is 'unmasked', and any further SEIs will be broadcast back to the failing OCP. SEI-Analyser therefore unsets MULT before reading the parameter.

Timeout system errors resulting from remote OCP or SAC accesses are not broadcast.

4.11 HARDWARE SYSTEM ERROR PARAMETER - 2960

System Error

Hardware System Error	(32 bit word)
Bits 0 - 2	Failing Module No. If the system error is reported in a processor other than the originating module, i.e. if bits 0-2 of the parameter are not equal to the module number of the reporting processor, bits 3-31 of the parameter are zero.
Bits 3, 4	01 Irrecoverable Error 10 Successful Retry 11 Retry Failed (Code 00 used for Software System Error)
Bits 5 - 7	Zero
Bits 8 - 11	ACR
Bit 12	Zero
Bit 13	0 Photograph taken 1 No Photograph
Bit 14	0 SSN+1 OCP Register Dump 1 No SSN+1 Dump
Bit 15	0 Full Photograph)valid only 1 Mini only)if bit 13 = 0
Bits 16 - 31	Signify the failing Sub-Module (if bits 3, 4 ≠ 0, 0)
Bit 16	SMAC system error including Multibit Hamming failure
Bit 17	Single bit (recovered) Hamming failure.
Bit 18	Decoder Parity Error
Bit 19	SAU Parity Error
Bit 20	Store time out
Bit 21	OCP Internal time out
Bit 22	Microprogram Engine Parity
Bit 23	Data Parity Error from SMAC
Bit 24	Microprogram detected error
Bits 25 - 31	

Other bits to be defined for internal OCP-detected failure Bits 16, 17 and 20 are used for the same purpose on 2970.

4.12 SOFTWARE SYSTEM ERROR PARAMETER - 2960

(32 bit word)

Bits 0 - 2	Module Detecting failure
Bits 3, 4	0, 0 Software System Error
Bits 5 - 7	Zero
Bits 8 - 11	ACR
Bits 12 - 15	Zero
Bit 16	Illegal Virtual Store condition
Bit 17	Class 5 Interrupt Masked (VSI)
Bit 18	Class 7 Interrupt Masked (PEI)
Bit 19	Class 8 Interrupt Masked (System Call)
Bit 20	Class 9 Interrupt Masked (Out)
Bit 21	Class 10 Interrupt Masked (Extra code)
Bit 22	Stack Segment Number is ODD
Bit 23	Segment Table Format Error
Bit 24	Entry Forced by Software (Image Store)
Bit 25	ACS = Ø on Initial Activate
Bits 26 - 31	-

N.B. 4014000F indicates a forced dump.

5.4.3 cont. 2976 types: Set FSEI (Frame 1, Row 3, Bit 17) using the 'continuous set' key. This causes an SGSE to be broadcast. Either OCP may be used.

6. IMAGE STORE

This section explains how and when VME/B manipulates the more important image store locations when configuring IN/OFF an OCP.

6.1 Image Store Format (2960/2970)

0 2	3 31
BLOCK NO	FORMAT IS BLOCK DEPENDENT

If bits 0 - 2 = 0 the the IS access is to this OCP

If bits 0 - 2 = 2 then the IS access can be to this OCP or any other Port.

If bits 0 - 2 in the image store address decode to 2 then the following address format applies for OCP registers. (For a full explanation see PSD 4.2.3G).

0 3	4 7	8 11	12 15	16 31
0 1 0 0	0 0 1 0	PORT	NOTE A	CPU REGISTER ADDRESS

N.B. 1 Bits 5 & 7 are undefined but are invariably zero.

N.B. 2 Bits 0 - 3 are altered by the CPU hardware before being passed on to the external destination.

PORT This is the port to which the IS Command is to be sent.

NOTE A Bits 12 - 15 are used as follows:

Bit 12 = 0 Do not suspend OCP on completion of IS Command

= 1 Suspend OCP on completion of IS Command (OCP commonly referred to as asleep).

Bit 13 - 0 Then bits 14 and 15 are ingnored.

= 1 Interrupt as follows:

6.1 cont

Bit 14 - 0 Cause MPI on completion of IS Command

= 1 Cause SEI on completion of IS Command

Bit 15 This bit defines where the MPI/SEI caused by Bit 14 is to be sent.

= 0 The interrupt is sent to the port other than the one specified in the IS Command and it is generated as if it is an External Interrupt, i.e. an IS location must be read to clear it.

= 1 The interrupt is sent to the port specified in the IS Command and it is generated as if it is an Internal Interrupt, i.e. entering the software interrupt procedure via the IST clears the interrupt.

N.B. Bits 14 and 15 determine the type and destination of interrupt to be generated after the IS access to the specified address has been completed.

6.2 CPU REGISTERS (2960/2970)

6.2.1 6009 BROADCAST SYSTEM ERROR

This register controls whether System Errors are to be broadcast to the other OCP or not.

Bit 31 = 1 Broadcast SE
= 0 Don't broadcast SE

In normal dual running Bit 31 is set in both OCPs so that when either OCP SE's it is broadcast to the other OCP. On receipt of a SE from the other OCP bit 31 of 6009 (in the OCP handling the system error) is cleared by hardware. Bit 31 can be reset by software after SEI-ANALYSER has decided whether it is going to carry on using the failing OCP.

Broadcast System Errors is turned on in the OCP being configured in by KRM-DUAL-OCP-TEST and in the other OCP by KRM-CONFIGURE-IN-OCP.

It is turned off either by hardware or KRM-CONFIGURE--OFF-OCP.

6.2.2 600A SYSTEM INTERRUPT MASK

This register controls communications between port connected devices.

Bits 24 - 27 = SA (ACTIVATES) = MBI bits 8-11 on 2960
28 - 31 = SI (INTERRUPTS) = MBI bits 12-15 on 2960

Bits 24 & 28 Control Port 0
25 & 29 Control Port 1 etc.

If a bit is set in SA then it inhibits activates being sent to that port.

If a bit is set in SI it inhibits interrupts from that port.

On a dual system the following applies: the IPL OCP can receive interrupts from and send activates to all ports, the non IPL OCP cannot receive interrupts from the SAC(s) on the system but can send activates to all ports. The reason the non-IPL OCP is closed to the SAC(s) is so that on peripheral interrupts both OCPs don't go for the PERM CPU SEMAPHORE resulting in one OCP servicing the interrupt and the other looping on the semaphore (when the looping OCP gets the semaphore it would fall through PERM-SERVICE-RESPONSE as the interrupt has been serviced).

The bit settings of SA/SI corresponding to the port number of this OCP are ignored.

i.e. for port 3 bits 27 & 31 are irrelevant
port 2 bits 26 & 30 are irrelevant

6.2.3 6012 LOAD MICROPROGRAM

This register controls loading the microprogram to the OCP being configured in. KRM-CONFIGURE-IN-OCP calls KRM-GIVE-MICROPROGRAM to get the OCP microprogram and place it in SMAC Ø BLOCK Ø. Whilst the OCP microprogram is in block Ø photographs are inhibited by writing to IS 6011.

6.2.4 6302 SYSTEM ERROR INTERRUPT

This register is read by SEI ANALYSER (in the OCP that had a SE) in order to clear down the SE.

When KRM-DUAL-OCP-TEST is running in the OCP being configured in it reads this register so that the OCP is not declared to the system with a system error pending.

6.2.5 6303 MULTIPROCESSOR INTERRUPT

This register is read by KRM-HANDLE-MPI or VMM-MULTI-OCP-INT-HANDLER to clear down MPI's generated on successful Configure-In or Process-Reschedule respectively.

6.2.6 2960 M.B.I. (w.s. = 16)

0	EVENT PEND
1	
2	
3	
4 } 5	1900 DEC.SLOT.No.
6 } 7	S4 DEC.SLOT No.
8-11	SI MASK INHIBIT PORTS 0-3 TX Activate
12-15	SI MASK INHIBIT PORTS 0-3 RX Interrupts
16	JUMP CONDITION TRUE
17	DO NOT USE µ-PROG SYS.CALL CODE.
20	IST ACCESS
21	APSTB (WD's 0,1 SMAC 1)
22	SSN+1 DUMP
23	BROADCAST SEI
24	MPI
25	SEI
26	EXIS
27	COMMUNICATION SMAC No.
28 } 29	CLOCK MODULE No.

6.3 2976 Types OCP Image Store

6.3.1 4012 - S1

Bits Ø to 3		Allow external interrupts from modules Ø to 3, e.g. Real Time Clock (See note 2)
4 to 5		ACK/NACK Synchronisation Stats.
6	PS2U	Alternative PSTB in use. Set when PSTB has been loaded from the alternative area in SMAC 1.
7	IPHOT	Inhibit Photodump.
8 to 11		Hardware use. See Note 1.
12 & 13		Unused.
14, 15		Inhibit inter-OCP communication from Modules 2 and 3 i.e. Allow activates when set to '0'. This is to allow communication (e.g. Remote IPL) to to an OCP in the reset state.
16 to 19		Allow system error interrupts from Modules Ø to 3. See Note 2.
22	MPIS	Cause multiprocessor interrupt to self.
23	MPIX	Cause multiprocessor interrupt to other OCP .
24	DUAL	OCP is capable of being dualled. Setting bit has no effect on OCP without DUAL mods.
25	FMDU	FMDU & MLIB fitted. This bit may be reset by software to inhibit iterative divide. Setting the bit has no effect on an OCP without FMDU.
26 to 29		Allow Multiprocessor and Peripheral interrupts from Modules Ø to 3. See Note 2.

N.B. Except for bits 14 and 15, Ø = inhibit and 1 =allow.

6.3.2 4013 - S2

Bits Ø to 7		Unused. See Note 1.
8 to 11		Real Time Clock module number. See Note 3.
12	NONOV	Non-Overlapped.

6.3.2 cont.

Bits 13 INHERET Inhibit retry.

14 INHWBD Inhibit wait before dump.

15 PS2A Set by software to indicate that the alternative PSTB area in SMAC 1 has been loaded. When set, causes system errors caused by SMAC Ø to use SMAC 1. If not set causes all system errors to use SMAC Ø.

16 MULT Set by software to indicate that this OCP is part of a multiprocessor.

It also has the effect of making the photodump area dependant on the OCP port no.

17 FSEI Causes a software system error if set to 1.

18 INHUB Inhibit use bits.

19 HOOT The hooter emits a continuous 1 Kc note when this bit is set to 1.

20 FSSNEQ Inhibit stack slaving.

21 LSSREJ Lock stack slave rejection pointer.

22 STCL Stop clock and initiate DTU.

23 IRETLOG Inhibit retry logging interrupt.

24 LISREJ Lock instruction slave rejection pointe

25 IACLOG Inhibit ATU/Cache Logging Interrupt for successful retry.

26 Unused.

27 LOPREJ Lock OP slave rejection pointer.

28 INHOP Inhibit OP slaving.

29 DD (Double Dump) Makes photodump area port dependent.

Note 1.

Those bits marked Unused or Hardware Use may be written to without causing a malfunction of the hardware.

Note 2.

These fields are bit significant with the lowest numbered bit corresponding to the lowest numbered module. If two Activates or two Interrupts occur at the same time priority is always given to the lowest numbered module.

6.3.2 cont Note 3.

4 Port SMAC

Module Number	Ø	1	2	3
Port Number	Ø	1	2	3
Geographical Processor Number (C = CPU, S = SAC)	S1	SØ	CØ	C1
PI Value in SSR	(-)	(-)	2	3

6.3.3 Inter-OCP Communication

To communicate with the other OCP, Load or Store ACC with image store format 42PØØØØM is used where P is the Port No. and M is the message as defined below.

M = Ø	Clear Slaves	
1	Remote IPL	IS Write
2	Remote register load	
3	Send Status	IS Read
4	Suspend	
5	Restart	
6	Clear MPI	IS Write
7	Illegal	

The contents of ACC (which must be single length) will be written to word 9 of the communication area, but will be ignored by the receiving OCP. Use of store B will be undefined.

Using an Image store write with message M = 3 or an Image store read with M = Ø, 1, 2, 4, 5, 6 or 7 is undefined, but will probably result in a lock up situation resulting in a System Error Timeout, or Post Timeout on subsequent use of the SMAC communication area.

Sending SAC image store (i.e. 44PØXXXX or 4ØPTXXXX) with an OCP port no. will result in a NACK from the receiving OCP, causing a Program Error in the sending OCP.

Using a message value M = 7 is undefined.

Setting P = Port No. of the sending OCP will cause a System Error Timeout.

6.3.3.1 Clear Slaves

This causes the slaves of the processor to be cleared in an orderly manner. The processor then continues processing. Acknowledge is sent to the initiating OCP after the slave stores have been cleared.

6.3.3.2 Remote IPL

This causes the processor to perform the pseudo-activate sequence which is normally started by the interrupt at the end of the IPL peripheral transfer. It is used in multiprocessors to bring the other OCPs into the system after the first OCP has been loaded.

Acknowledgement is sent to the other OCP before the IPL takes place.

6.3.3.3. Remote Register Load

This has the same effect as Remote IPL but the AU microprogram load is inhibited. Acknowledgement is sent to the other OCP before the register load takes place.

6.3.3.4 Suspend

This causes the processor to be suspended.

6.3.3.5 Restart

Used to restart the processor after a Suspend message, or after a System Error Interrupt.

6.3.3.6 Clear MPI

This causes the external multiprocessors interrupt bit MPIX to be cleared.

6.3.3.7 Send Status and Clear SEI

The status which corresponds to the parameter stacked on system error is read into A3, and the system error interrupt cleared.

This instruction is only meaningful if the interrogated OCP has signalled a system error and has suspended, and will always leave that OCP suspended.

Note that the system error parameter is also dumped on System Error.

6.3.3.8 Illegal Messages

An illegal combination of bits in bits 29 - 31 (i.e. 7) or bit 17 set, or SAC type image store to an OCP port (i.e. bit 6) will result in a NACK causing the sending OCP to program error, and will not interfere with the receiving OCP which will continue with normal instruction sequencing.

6.3.4 Multiprocessor Interrupt

Two bits in the configuration register can be set by software to cause a multiprocessor interrupt in either OCP.

S1 bit 22 - MPIS
causes an MPI to the sending OCP only.

S1 bit 23 - MPIX
causes an MPI to the other OCP only.

MPIX will normally be cleared by Inter processor image store "Clear MPI" from the other OCP.

MPIS will normally be cleared by image store write Ø to bit 22 of the configuration register. In order for an MPI to occur, Allow Normal Interrupts must be set in the receiving OCP i.e. for the case of MPIS the OCPs own port number must be set.

6.3.5. Broadcast System Error (SEIO)

If this bit is set, any incoming inter-OCP Image Store that normally allows the OCP to continue processing i.e. Clear Slaves, Restart, Clear MPI and Illegal Message will leave the OCP suspended. The SEIO must be cleared by "Send Status and clear SEI" before the OCP can be restarted. "Remote IPL" and "Remote Register Load" do not require this bit to be reset in order to work.

6.4. SMAC REGISTERS

6.4.1 4A20) SMAC CONFIGURATION REGISTER
6A20)

This register is written to in each SMAC by KRM-CONFIGURE-IN/OFF-OCP to unset/set bits 26 - 29. Bits 26 - 29 control whether that port is allowed to talk to that SMAC.

1 = closed to that port.

0 = open to that port.

6.5 SAC REGISTERS

6.5.1 4500) 6500) SAC LOWER WINDOW

This register is read by KRM-DUAL-OCP-TEST in each SAC known to the other OCP. It is read to ensure that the OCP being configured in can access the SAC(s) known to the first OCP. It it can't then the result code KRM-MPROG-LOAD-FAILS is returned.

7.1 To Configure an OCP In

(a) Ensure that the OCP is on 'run', and for 2960 that the TLOE cassette is in the ECP reader.

(b) On the SRU

If both OCPs are currently in the same configuration:-

1. Depress the appropriate change key.
2. Press enable and return. The OCP will now be 'free'.
3. Press enable and return. This will reset the OCP by sending a local IPL.
4. Reset the change key. The OCP will now be back in the configuration.

If the OCPs are in different configurations then to move OCP2 from configuration 'B' to configuration 'A':-

1. Depress the change key for OCP2 in configuration 'B'.
2. Press enable and return on configuration 'B'.
3. Reset change key. The OCP will now be free.
4. Depress the change key for OCP2 in configuration 'A'.
5. Press enable and return on configuration 'A'. This will reset OCP2 by sending a local IPL.
6. Reset the change key. OCP2 will now be in configuration 'A'.

(c) On 2960, wait until the status 'stopped' light goes out, indicating that IPLM has loaded off the cassette.

(d) CDS OCPn, IN.

4.11 HARDWARE SYSTEM ERROR PARAMETER - 2960

System Error

Hardware System Error	(32 bit word)
Bits 0 - 2	Failing Module No. If the system error is reported in a processor other than the originating module, i.e. if bits 0-2 of the parameter are not equal to the module number of the reporting processor, bits 3-31 of the parameter are zero.
Bits 3, 4	01 Irrecoverable Error 10 Successful Retry 11 Retry Failed (Code 00 used for Software System Error)
Bits 5 - 7	Zero
Bits 8 - 11	ACR
Bit 12	Zero
Bit 13	0 Photograph taken 1 No Photograph
Bit 14	0 SSN+1 OCP Register Dump 1 No SSN+1 Dump
Bit 15	0 Full Photograph)valid only 1 Mini only)if bit 13 = 0
Bits 16 - 31	Signify the failing Sub-Module (if bits 3, 4 ≠ 0, 0)
Bit 16	SMAC system error including Multibit Hamming failure
Bit 17	Single bit (recovered) Hamming failure.
Bit 18	Decoder Parity Error
Bit 19	SAU Parity Error
Bit 20	Store time out
Bit 21	OCP Internal time out
Bit 22	Microprogram Engine Parity
Bit 23	Data Parity Error from SMAC
Bit 24	Microprogram detected error

Bits 25 - 31

Other bits to be defined for internal OCP-detected failure Bits 16, 17 and 20 are used for the same purpose on 2970.

4.12 SOFTWARE SYSTEM ERROR PARAMETER - 2960

(32 bit word)

Bits 0 - 2	Module Detecting failure
Bits 3, 4	0, 0 Software System Error
Bits 5 - 7	Zero
Bits 8 - 11	ACR
Bits 12 - 15	Zero
Bit 16	Illegal Virtual Store condition
Bit 17	Class 5 Interrupt Masked (VSI)
Bit 18	Class 7 Interrupt Masked (PEI)
Bit 19	Class 8 Interrupt Masked (System Call)
Bit 20	Class 9 Interrupt Masked (Out)
Bit 21	Class 10 Interrupt Masked (Extra code)
Bit 22	Stack Segment Number is ODD
Bit 23	Segment Table Format Error
Bit 24	Entry Forced by Software (Image Store)
Bit 25	ACS = Ø on Initial Activate
Bits 26 - 31	-

N.B. 4014000F indicates a forced dump.

5.2 Layout of KEM-Store- Allocation

5.2.1 2960

Real Address (bytes)

HEX	DECIMAL	
4C-5F	76-95	Software System Error parameter
60-63	96-99	Hardware stacked SEI parameter
64	100	Type of unit (HM_OCP) = X'91'
65	101	PORT number
66-67	102-103	The length of the rest of the OCP part of the log
68-7F	104-127	Unused
80-BF	128-191	Copy of SSN+1 registers
C0-C7	192-199	LSTBØ, 1
C8-CF	200-207	PSTBØ, 1
D0-FF	208-255	Unused
100-1FF	104-511	The OCP Photograph for PORT 2
200-2FF	512-767	The OCP Photograph for PORT 3

The next section will be variable in length depending on the number of SMACs. For each SMAC in the system, the following 20 byte areas will be produced:-

Ø	Type of Unit (HM_SMAC) = X'92'
1	SMAC number
2-3	Size of rest of this entry (16)
4-7	Failing data for this SMAC (4COS6004)
8-11	Failing address for this SMAC (4COS6100)
12-15	Failing status for this SMAC (4COS6A00)
16-19	Configuration for this SMAC (4COS6A20)

6.1 cont

Bit 14 - 0 Cause MPI on completion of IS Command

= 1 Cause SEI on completion of IS Command

Bit 15 This bit defines where the MPI/SEI caused by Bit 14 is to be sent.

= 0 The interrupt is sent to the port other than the one specified in the IS Command and it is generated as if it is an External Interrupt, i.e. an IS location must be read to clear it.

= 1 The interrupt is sent to the port specified in the IS Command and it is generated as if it is an Internal Interrupt, i.e. entering the software interrupt procedure via the IST clears the interrupt.

N.B. Bits 14 and 15 determine the type and destination of interrupt to be generated after the IS access to the specified address has been completed.

6.2 CPU REGISTERS (2960/2970)

6.2.1 6009 BROADCAST SYSTEM ERROR

This register controls whether System Errors are to be broadcast to the other OCP or not.

Bit 31 = 1 Broadcast SE
= 0 Don't broadcast SE

In normal dual running Bit 31 is set in both OCPs so that when either OCP SE's it is broadcast to the other OCP. On receipt of a SE from the other OCP bit 31 of 6009 (in the OCP handling the system error) is cleared by hardware. Bit 31 can be reset by software after SEI-ANALYSER has decided whether it is going to carry on using the failing OCP.

Broadcast System Errors is turned on in the OCP being configured in by KRM-DUAL-OCP-TEST and in the other OCP by KRM-CONFIGURE-IN-OCP.

It is turned off either by hardware or KRM-CONFIGURE--OFF-OCP.

6.2.2 600A SYSTEM INTERRUPT MASK

This register controls communications between port connected devices.

Bits 24 - 27 = SA (ACTIVATES) = MBI bits 8-11 on 2960
28 - 31 = SI (INTERRUPTS) = MBI bits 12-15 on 2960

Bits 24 & 28 Control Port 0
25 & 29 Control Port 1 etc.

If a bit is set in SA then it inhibits activates being sent to that port.

If a bit is set in SI it inhibits interrupts from that port.

On a dual system the following applies: the IPL OCP can receive interrupts from and send activates to all ports, the non IPL OCP cannot receive interrupts from the SAC(s) on the system but can send activates to all ports. The reason the non-IPL OCP is closed to the SAC(s) is so that on peripheral interrupts both OCPs don't go for the PERM CPU SEMAPHORE resulting in one OCP servicing the interrupt and the other looping on the semaphore (when the looping OCP gets the semaphore it would fall through PERM-SERVICE-RESPONSE as the interrupt has been serviced).

The bit settings of SA/SI corresponding to the port number of this OCP are ignored.

i.e. for port 3 bits 27 & 31 are irrelevant
port 2 bits 26 & 30 are irrelevant

6.2.3 6012 LOAD MICROPROGRAM

This register controls loading the microprogram to the OCP being configured in. KRM-CONFIGURE-IN-OCP calls KRM-GIVE-MICROPROGRAM to get the OCP microprogram and place it in SMAC Ø BLOCK Ø. Whilst the OCP microprogram is in block Ø photographs are inhibited by writing to IS 6011.

6.2.4 6302 SYSTEM ERROR INTERRUPT

This register is read by SEI ANALYSER (in the OCP that had a SE) in order to clear down the SE.

When KRM-DUAL-OCP-TEST is running in the OCP being configured in it reads this register so that the OCP is not declared to the system with a system error pending.

6.2.5 6303 MULTIPROCESSOR INTERRUPT

This register is read by KRM-HANDLE-MPI or VMM-MULTI-OCP-INT-HANDLER to clear down MPI's generated on successful Configure-In or Process-Reschedule respectively.

6.2.6 2960 M.B.I. (w.s. = 16)

0	EVENT PEND
1	
2	
3	
4, 5	1900 DEC.SLOT.No.
6, 7	S4 DEC.SLOT No.
8-11	SI MASK INHIBIT PORTS 0-3 TX Activate
12-15	SI MASK INHIBIT PORTS 0-3 RX Interrupts
16	JUMP CONDITION TRUE
17	DO NOT USE μ-PROG SYS.CALL CODE.
20	IST ACCESS
21	APSTB (WD's 0,1 SMAC 1)
22	SSN+1 DUMP
23	BROADCAST SEI
24	MPI
25	SEI
26	EXIS
27	COMMUNICATION SMAC No.
28, 29	CLOCK MODULE No.

HELP

Should the HELP line be pulled by a bit being set in the HELP register (see 3.6.1) then the resulting action is shown in the flowchart, on the following page.

The flowchart refers to the options set in WS 27 whose contents can be set as follows:-

WS #27			Set by
BITS	0 - 14	Spare	
15	-	Error handling in progress	ECP during photo procedure
16 - 19	-	Reserved	
17	-	Stop after photo	*Type "HELP" on ECP
20	-	Stop on SEI	System Software or *Type "HELP OFF" on ECP
21	-	Stop on PEI	Syst. Soft/Operator
22	-	Photo on PEI	Syst. Soft/Operator
23	-	Reserved	
24	-	Inhibit Hamming Correction Reporting.	Syst. Soft/Micro prog.
25 - 27	-	Reserved	
28	-	Inhibit data slave	
29 - 30		Reserved	
30 —		ATS.	
31 = 1 ⧈		Inhibit full photo (for Soft or External SE-No Photo) (for P.E. and u-prog detected errors - Mini photo).	Syst. Soft.

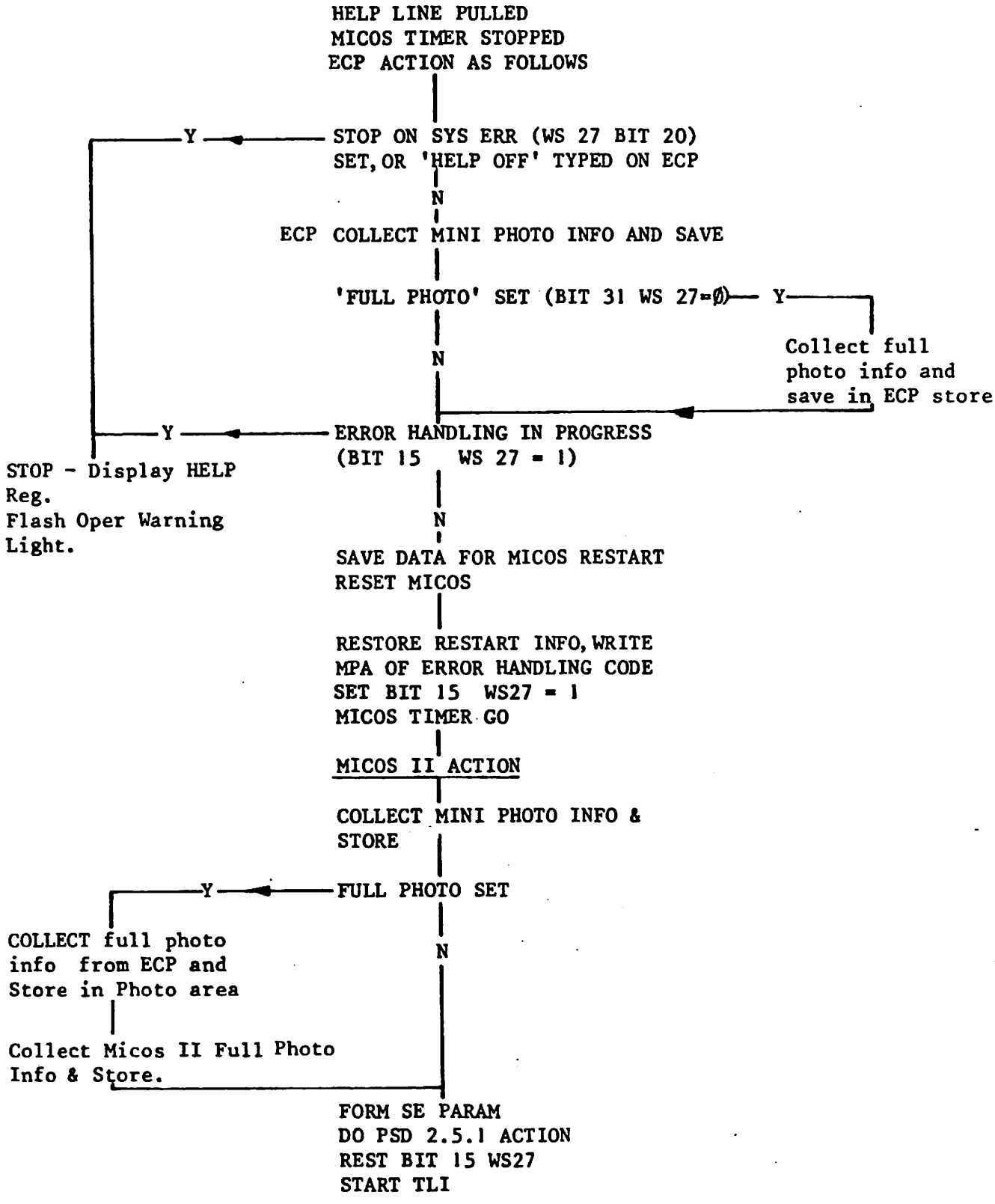
ECP COMMANDS:-

* "HELP" - Takes immediate photograph and halts.

"HELP OFF" - HELP Reg. displayed and halts. (No photo).

"HELP ON" - Restores Normal handling.

HELP HANDLING Various parts of photo collected by ECP or MICOS



IMP

IMP Call Sequences

The pre-call sequence is first to odd-align the stack front and then to store the current LNB at SF (this SF to be the new LNB, i.e. in the called routine). The stack front is further raised by 4 words for the link descriptor (placed in old LNB+1,2 by the CALL) and for the called routine's PLT descriptor, to be placed in LNB+3,4 by the post-call sequence. The purpose of odd-aligning the stack front is so that the link and PLT descriptors are double word aligned. This is all done by the single instruction

```
PRCL    4
```

Next the parameters are stacked (if required) and LNB is raised by 5 plus the number of parameter words to point to the new display (starting at the stack front as it was after the odd-aligning but before the storing of LNB).

Programmers should note the effect of the pre-call alignment when designing routine headings. For example

```
routine ONE(integer A, integername B)
```

results in the descriptor to B being across a double-word boundary and requires 2 core cycles to fetch it.

```
routine TWO(integername B, integer A)
```

is more efficient.

XNB is set to the called routine's textually surrounding routine for internal calls.

Finally the CALL is executed, with operand either

(i) for an internal routine: N	where N is offset in halfwords from current PC
(ii) for an external routine: @(XNB+N)	where XNB+N has a descriptor descriptor which points to the code descriptor in the called routine's PLT.

The PLT is conceptually an array of code descriptors for the routines in the module. An external reference is a descriptor descriptor to one of these (i.e. a descriptor which points to one of these). The loader fills in the addresses appropriately.

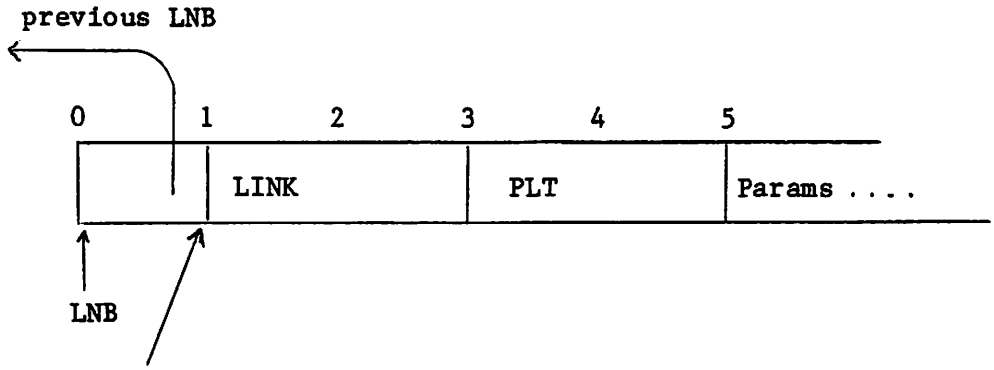
To summarise, the CALL instruction loads DR with the operand descriptor (if the operand is indirect). It also places a link descriptor at old LNB+1,2. LNB has been set up by the pre-call sequence to point to the new display.

For external routines, the post-call sequence (in the called routine) is to store DR, which has a descriptor to the descriptor in the called routine's PLT, at LNB+3,4, having used INCA to set the address in DR back to the start of PLT. That is, LNB+3,4 have a descriptor to this routine's PLT.

So far,

INCA	-N	where N is the offset of this routine's descriptor from the start of PLT (omitted if N=0).
LDB	0	set bound to zero now, but later set to offset from CST of ON and diagnostic block patched (ORed) in at end of routine.
STD	(LNB+3)	and 4.

LNB+4 has the address therefore of PLT (GLA), and this is loaded to XNB subsequently as required (e.g. for referencing <u>own</u> variables and for making further copies).



The last bit of the stored LNB is used to indicate that SF was increased by 1 in the pre-call sequence - it should be ignored when chaining back in dumps, etc.

P.D. Stephens

%NAME Parameters in IMP

The general parameter %NAME allows any entity to be passed to the called routine. This facility is used in the sequential and direct access routines but is not a user facility. It is possible to find out quite a lot about the actual parameter by examining the descriptor.

For variables as actual parameters the descriptor is a scaled vector descriptor with the address field pointing at the leftmost byte of the actual parameter. The size of the item can be obtained from the size code of the descriptor, as follows:

3 = byte
4 = half (care needed with this - not supported by hardware)
5 = integer or real (32-bit)
6 = long integer or long real (64-bit)
7 = long long real (128-bit)

The bound field gives the type as follows:

1 = integer
2 = real
3 = Boolean (ALGOL(E) only)

For structures (i.e. strings and records) an unscaled byte descriptor is passed with the bound field containing the size of the entity in bytes.

Having decoded the descriptor, the program can store items back into the variable using IMP mapping functions; i.e.

```
BYTEINTEGER(ADDR(name)) = ...

STRING(ADDR(name)) = ...
```

as appropriate.

P.D. Stephens

EMAS 2900 IMP NOTE

No: 3
Date: 10/11/78

Extra Facilities in Release 8

Release 8 of IMP has some extra facilities on an experimental basis - they may be withdrawn at any time. To use the compiler, insert ERCC07.NRIMP08Y in your directory and log out.

1) Profiling

If a file is compiled using PARM(PROFILE), extra code is inserted to maintain a count of how often each (executable) line is executed. If there are several statements on a line each will increment the count. A call on the special intrinsic routine PPROFILE will print out details of the most heavily used lines and reset all counts. Programs being run on other than the standard Subsystem will require to be linked with ERCC07.PPROF01Y to obtain special service routines for the profiling.

N.B. Profiling is very demanding of CPU time - expect programs to run at least 3 times slower.

2) Conditional compilation

If an IMP statement starts with ? it will be analysed but no code will be generated. However if PARM(PARMX) is specified then code for such a line will be generated normally. This provides a convenient method of including additional diagnostic code in software which can be quickly incorporated at moments of crisis.

3) Constant shuffling

Normally IMP pools all numeric and string constants by building a constant table which is added to the back of the code. This minimises program size. However when compiling files of routines where one may be used extensively and the others ignored it may be better to incorporate each routine's constants with its code. This will give a larger program but fewer page turns. If PARM(PARMY) is selected then the latter method of constant inclusion is used. It is likely that Subsystems, Directors and a few large libraries will benefit from this option.

Feedback on the effects (if any!) would be welcome.

P.D. Stephens

Interactive Debugging in IMP

DEBUG is an interactive debugging aid for IMP programs on EMAS 2900. It includes commands for examining and modifying data and hence changing the course of a computation, for setting breakpoints, and for the conditional execution of DEBUG commands. The breakpoint scheme makes it possible for the user to gain control during program execution and then specify any further action he wishes. The conditional execution facility allows DEBUG commands specified by the user to be stored, and executed only when certain conditions related to the running program are satisfied, e.g. when a given line number in the program has been reached.

In general, the intention has been to allow the user to intervene in the execution of the running program in a way consistent with the effect that would be produced by additional source statements in the program. Hence it is only possible to examine or change variables which are in scope when the DEBUG command is executed. However conditional commands may be issued at any time regardless of whether any variables they refer to are currently in scope.

Establish access to DEBUG by first typing:

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)
```

Then compile the program with PARM(DEBUG) set. Additional parms may be set with the exception of NOTRACE, NODIAG and OPT.

When the program is run, DEBUG will print the current line number and then prompt for a command:

```
DEBUG n: (where n is the index number of the command - see Kill)
```

The user can then input one or more DEBUG commands. Commands take two forms, simple and conditional. Simple commands, which consist of a letter followed, if appropriate, by a parameter, are executed immediately and then discarded. Conditional commands are stored, and only executed when the specified condition is satisfied. The program will only resume execution when the command R (see below) is given.

Conditional commands take the form:

```
<simple command> .IF <conditional expr>
```

or, for typing convenience,

```
<simple command> @ <conditional expr>
```

The following simple commands are available:

A <variable>=<value>	- Assign the value to the variable; it is not possible to assign a value to an unequivalenced name type variable.

B - Set a breakpoint; when executed, accept a DEBUG command from the console. B is only used as part of a conditional command.

D <addr>,<length>,<format> - Dump an area of the virtual memory of <length> bytes, starting at <addr> to the console. <format> may be C for a character dump or H for a hexadecimal dump. The user must have determined the relevant virtual memory addresses prior to the execution of the program.

F - Print a map of the object file, indicating the starting line numbers of the routines and blocks it contains.

H - Halt execution and return immediately to command level.

I - Ignore all stored conditional commands and continue execution of the program in non-debug mode.

K <number> - Kill the specified stored commands. The parameter can include several command index numbers or ranges of command index numbers separated by commas, e.g.

```
K 2,4,6-8
```

Note that the command index number n is given in the prompt "DEBUG n:" current when the command was input.

M - Execute %MONITOR.

P <variable> - Print the value of the named variable.

R - Resume execution of the program. DEBUG will force a break at the next line unless at least one conditional command is currently stored.

S <line no>,<count> - Print <count> lines of the source listing starting at line <line no>. A prompt is issued for the name of the compiler generated listing file.

Three types of conditional expression are available; they may be used singly or in combination:

```
<simple command> @ <conditional expr>

or  <simple command> @ <conditional expr>&<conditional expr>

or  <simple command> @ <conditional expr>&<conditional expr>&<conditional
                                                                  expr>
```

Only one of each type can be specified in a single command. The three types are as follows:

C=<variable> - This is satisfied whenever the specified variable has its value changed. If the variable is reassigned its current value, this is not detected as a change.

R=<routine/block> - This is satisfied when the current line being executed lies within the specified routine or block. For blocks, the starting line number should be given; for routines, functions or maps, the name should be given. Note that this condition is not satisfied when the current line lies within a routine or block which itself lies within the specified routine or block.

L=<line spec> - This is satisfied when the current line being executed lies within the line spec. The line spec is defined below.

<line spec> : <line>[,<line spec>]

<line> : <line expr> ! <line range>

<line range> : <line no> - <line no>

<line expr> : +<line no> ! -<line no> ! *<line no> ! #<line no>

The use of these is explained below:

L=n - Satisfied when the current line number is n.

L=n-m - Satisfied whenever the current line number lies between n and m.

L=+n - Satisfied when the line number equals the current line number plus n (i.e. the line number current when the command was given + n)

L=-n - Satisfied when the line number equals the current line number minus n.

Commands involving these expressions are permanently stored. However two temporary forms are provided which cancel themselves once satisfied:

L=*n - Satisfied when the current line is encountered for the n'th time. This is intended for skipping through an iterative sequence without stopping at every iteration.

L=#n - Satisfied when the n'th executable line after the current line is encountered. This allows control to be returned to the user after a test and branch operation, the result of which is not known in advance.

Examples

Several examples of command lines follow:

DEBUG 3:P VAL @ C=VAL & R=COMBINE	- Print variable VAL, whenever its value changes within routine COMBINE.
DEBUG 1:A S="HEADING"	- Assign "HEADING" to string variable S.
DEBUG 4:B @ L=10-20 & C=JIM	- Break whenever the value of JIM changes and the current line lies between 10 and 20.
DEBUG 8:R	- Resume execution of the program.

Notes

1. A newline terminates a command line unless it is part of a string (i.e. preceded by a quote). Spaces are not significant unless enclosed in quotes.

2. The index number n printed in the prompt DEBUG n: is incremented only if a storable (i.e. conditional) command is specified. When commands are Killed, the index number is unchanged except for the case in which all commands are Killed, when its value returns to 1.

3. If more than one C=<variable> commands are given for the same variable, then the same R=<routine> expression (if used at all) must be used in every case. This ensures that a single copy of the variable's value is held and hence avoids inconsistencies.

4. All the simple commands, with the exception of K (kill) and F (file map), may be used with a conditional expression to form a conditional command.

5. Commands B (break) and R (resume) operate by setting a single switch. Hence if a B conditional command succeeds but is followed by an R conditional command which also succeeds, the effect of the B is lost and no break occurs.

6. Except in line specification, a number may either be expressed in decimal or hexadecimal form, e.g. 32 or X20.

7. At present, users should only DEBUG one object file at a time. Thus a program and an external routine which it calls must not both be compiled with PARM DEBUG.

8. When a command such as B @ C=JIM is given, the condition is evaluated every time a line of the program is executed. In this case the evaluation involves a search for a declaration of the variable JIM; if it is not found, this is treated as a fault. In addition if there is more than one declaration of JIM in a program (or in the line range specified), the first accessed is used subsequently and the second, whenever encountered, is treated as a fault. Hence to avoid numerous error messages it is best to give a qualifying L= or R= expression with any C= expression.

9. Variables of all types except records may be referred to in Assign and Print commands or C= conditions. However, array elements must be specified with constant subscripts.

10. For every line of a program compiled with PARM DEBUG, a call is planted to the DEBUG routine which, consequently, is called many times in the execution of the program. To minimise overheads it is advisable to reach that part of the user program which is of interest before setting up several stored commands. In particular, repetitive operations like the initialisation of arrays become very slow if several stored commands have their conditional expressions evaluated on every line executed. However if only a single command of the form B @ L=n is stored, this will incur minimum overheads.

Sandy Shaw

February 1982

IMP80 on EMAS 2900: Differences from IMP9

Contents

		Page
1.	Compiler name	2
2.	Lower case input	2
3.	Continuation	2
4.	Comments	2
5.	== and ##	3
6.	Available types	3
7.	Keyword and operator alternatives	3
8.	own initialisation	3
9.	Switch labels	4
10.	Cycles	5
11.	start/finish blocks	6
12.	Constants	7
13.	Strings	8
15.	Records	9
15.	external items	10
16.	Procedures as parameters	11

Introduction

This document is intended for users of the programming language IMP on EMAS 2900 who wish to know how the new version of IMP, IMP80, differs from the current version, IMP9.

It should be noted that IMP80 on EMAS 2900 differs in certain respects from other implementations of IMP80, and that this document should not be trusted as far as other implementations are concerned.

Some of the features of IMP80 described below exist in IMP9. They are included here either to help explain some other feature or for completeness.

1. The command invoking the compiler is IMP80, not IMP.

2. Except within single or double quotes, lower case text is not distinguished from upper case. Thus

```
%integer a      and      %INTEGER A
```

are both acceptable and treated as equivalent. Note that

```
%integer Item 1a
```

is not distinguished from

```
%INTEGER ITEM1A
```

The convention in this document is that IMP keywords are underlined and given in lower case, with identifiers in upper case. Thus:

```
integer A
```

3. Continuation of statements. Statements can be continued on the next line by terminating the current line with c. The c is not required if the break comes immediately after a comma. (This applies to all statement types, not just own array initialisations.)

Examples:

```
if A=23 and K<=14 then c
   L=17 and M=18

integer A, B, C, D, E,
        F, G, H, I, J
```

A blank line following a line terminated by c is ignored.

4. Comments. A semi-colon does not terminate a comment - it can only be terminated by a newline. Comment statements can be continued by use of c, OR BY BEING BROKEN AFTER A COMMA (see 3 above).

A new type of comment is introduced; it is delimited by curly brackets, '{' and '}'. Such a comment can appear between atoms of a statement (an atom is an identifier, constant, keyword, operator or delimiting symbol).

Example:

```
A(I{month}, J{salary}) = 927.4
```

The comment text can contain any symbols except '}' and newline. The closing '}' can be omitted, in which case the comment is terminated by the next newline.

{...} comments are particularly useful for explaining own array initialisations.

5. == and ## (or \==)

The == operator can be used in conditions:

Example:

```
if A == B then .......
```

The condition is only true if A and B refer to the same variable; i.e. address and type equivalence is required. The operator ## (or \==) can be used to express the inverse condition:

```
if A ## B then .......
```

Note that == and ## can only be used to compare references to scalar variables, not to arrays.

6. Available types

byte integer	)	
half integer	)	
integer	)	
long integer	)	all of these can be
real	)	followed by array or
long real	)	name or array name
long long real	)	
string (n)	)	
record (format)	)	

A half integer variable requires 16 bits (2 bytes) of storage. It holds an unsigned integer value, in the range 0-65535.

The statements reals long and reals normal are not available in IMP80.

7. Keyword and operator alternatives

```
! or | for comment
fn for function
const for constant
byte for byte integer
half for half integer
\ for ** (real exponentiation)
\\ for **** (integer exponentation)
<> or \= for #
~ for \ (logical 'not')
\== for ##
```

8. own initialisation

a) The statement

```
own integer A
```

declares an own integer variable A and initialises it to 0 (the default value when no value is specified).

The statement

```
own integer X, Y, Z=4
```

declares X, Y and Z and initialises them to 0, 0 and 4 respectively. In IMP9 this statement causes X, Y and Z to be set to 4, 4 and 4. Note the difference!

It is bad practice to rely on default initialisation values, especially in IMP80, where existing implementations do not have the same defaults. The statements above should have been given as

```
own integer A=0
own integer X=0, Y=0, Z=4
```

These are unambiguous, whichever version of IMP is used.

b) For convenience, constants used in own array initialisations can be followed by a repeat count, in brackets. This repeat count can be given as '(*)' where * represents the number of remaining array elements to be initialised.

Example:

```
own integer array VALUES (1:50) = %c
 17, 4, 6(3), 9, 22(17),
 100(*) {all the rest}
```

This also applies, of course, to constant and external array initialisation.

c) Own arrays can be multi-dimensional. As before, the bounds must be constants or constant expressions. The order in which array elements are assigned the initialising values is such that the first subscript changes fastest. Thus, for an array A(1:2,1:3), the order of assignment would be A(1,1), A(2,1), A(1,2), A(2,2), A(1,3), A(2,3).

9. Switch labels

Consider the following:

```
switch LETTER('a':'z')
 :
 :
LETTER('a'):
LETTER('e'):
LETTER('i'):
LETTER('o'):
LETTER('u'):
! Deal with the vowels here
 :
LETTER(*):
! All the rest (i.e. the consonants)
 :
```

Instead of using a constant to specify a specific element of a switch vector, * can be used. It represents all the elements of the switch vector not defined elsewhere. Note that it does not have to come after the specifically defined switch labels.

10. Cycles

The permissible forms of cycle are these:

a) cycle (endless cycle)
:
repeat

b) while condition cycle
:
repeat

c) cycle
:
repeat until condition

d) for var = init, inc, final cycle
:
repeat

The unconditional instructions continue and exit can be used inside a cycle of any type. continue causes a branch to the next repeat; exit causes a branch to the statement following the next repeat.

Notes on the cycle types:

b) while cycles are executed zero or more times. When the cycle body consists of a single statement, the form

statement while condition

can be used.

Example:
SKIP SYMBOL while NEXT SYMBOL=' '

The IMP9 form while condition then statement is not allowed.

c) until cycles are executed one or more times. The simple form is

statement until condition

The IMP9 form until condition then statement is not allowed.

d) for cycles: the cycle variable must be of type integer; it should not be changed explicitly within the cycle body; (final-init) must be exactly divisible by inc; the cycle body is executed (final-init)//inc + 1 times or zero times, whichever is the greater; if the cycle body is not executed the cycle variable is set to be unassigned. It follows from this that a cycle starting

for I=10,1,8 cycle

will not be executed, but it will not be faulted either. This differs from IMP9, where the equivalent form

cycle I=10,1,8

would be faulted.

The simple form of for is

```
statement for var = init, inc, final
```

Example:

```
A(I)=0 for I=20,-1,1
```

[Going down in steps of -1 to 1 happens to be more efficient on EMAS 2900 than the more usual 1,1,20 form.]

11. start/finish blocks

The general form is

```
if cond 1 then start
      :
      :
      :
finish else if cond 2 then start
      :
      :
      :
      :
finish else if cond n then start
      :
      :
      :
finish else start
      :
      :
      :
finish
```

Notes

* Every start matches with the next occurring finish. If they enclose only one statement then they can be replaced by that statement.
  Example:

```
if cond 3 then start
  statement
finish else if .......
```

  can be expressed as

```
if cond 3 then statement else if ......
```

* then start can be replaced by start.

* if can be replaced by unless, the effect being to negate the condition following.

* Any of the statements starting "finish else" in the general form can be omitted, including the last one.

* If the condition controlling a start/finish block can be determined at compile-time then the IMP80 compiler may do so, and might not generate code for statements that cannot logically be executed. This is known as "conditional compilation".

12. Constants

a) An integer constant of any integer base from 2 to 36 may be specified. The form is

base_constant

where base is a decimal value and constant is an integer expressed with respect to the base. The letters A, B, ..., Y, Z can be used to represent the digits 10, 11, ..., 34, 35 in the integer.

Examples:

2_1010	ten in binary
8_12	ten in octal
16_A	ten in hexadecimal

An alternative form is provided for binary, octal and hexadecimal constants:

B'1010'	ten in binary
K'12'	ten in octal
X'A'	ten in hexadecimal

b) Named constants

Variables of all types can be given the attribute constant. This can be considered a special form of own variable, which cannot be changed from its initial value. However it is probably better to consider such variables as "named constants", since 1) this accords with their intended use, i.e. for replacing arithmetic or string constants within code by meaningful names; and 2) they do not have addresses, unlike other variables (but like constants).

Wherever a constant is permitted in an IMP80 program, a "constant expression" can be used instead. A constant expression is one which can be evaluated at compile-time, i.e. its operands are constants or named constants.

Example:

```
string (73) DELIVERY
```

can be replaced by

```
constant integer MAXNAME=20, MAXADDRESS=52
string (MAXNAME+1{for the newline}+MAXADDRESS) DELIVERY
```

Example:

```
constant integer NO=0, YES=1,
                 INPUT=1, CALCULATION=2,
                 OUTPUT=3
switch PHASE(INPUT:OUTPUT)
  :
  :
->PHASE(OUTPUT) if DONE=YES
  :
  :
PHASE(OUTPUT): ! Now print the results
  :
```

13. Strings

a) The keyword string may always be followed by a length specification.

Thus `string(10)array name ....`

and `string(255)name ......`

are permitted.

In EMAS 2900 IMP80, no use is made of the maximum length specification for string name and string array name variables.

[In other IMP80 implementations, however, a string name variable must have a maximum length specification and can only refer to ("be pointed at") a string variable of the same maximum length. The forms

```
string(*)array name .....
string(*)name ...........
```

are also provided, however, to enable declarations of reference variables which can point at any string variable.]

b) The string function FROMSTRING is renamed SUBSTRING.

c) A string resolution of the form

```
S -> (A).B
```

succeeds in IMP9 only if string S starts with string expression A. In IMP80, however, the resolution is interpreted as being equivalent to S -> JUNK.(A).B where JUNK is a "hidden" string (255) variable; that is, the resolution will succeed if A appears anywhere within S.

When converting an IMP9 program to IMP80, the following translation is recommended:

```
if S -> (B).C then ...                                  in IMP9
```

becomes `if S -> NS1.(B).NS2 and NS1="" then C=NS2 and ...` in IMP80

[NS1 and NS2 are new string (255) variables]

This translation is still valid when the IMP9 statement is

```
if S -> (B).S
```

i.e. when C is S.

Unconditional resolutions can normally remain unchanged; they might succeed in IMP80 where they would fail in IMP9, but this is not significant unless you are expecting them to fail.

14. Records

a) The syntax of declarations in IMP80 differ from those in IMP9. They are of the form

```
record (format) ident, ...
record (format) array ident, ...
record (format) name ident, ...
record (format) array name ident, ...
```

"format" is either the name of a record format previously described, the name of a record previously declared, or the actual record format itself.

Example:

```
record format RF(integer I, J, K)
record (RF) R
```

and

```
record (integer I, J, K) R
```

are both valid and have the same effect, except that the first version declares a record format with identifier RF, which can be used elsewhere, clash with other identifiers, etc. Either of the above forms could be followed by the statement

```
record (R) P
```

which would declare a record with the same format as that of record R.

To summarise: the keyword record in IMP80 must be followed by the keyword format or by a bracketed format or format reference or record reference.

[This syntax change can cause difficulties when translating IMP9 programs: a routine spec such as

```
routine spec NAME1(record name NAME2, ...)
```

must now be converted to

```
routine spec NAME1(record (FORM2) name NAME2, ...)
```

The record format FORM2 is presumably declared somewhere in the program, since a record of this format is required in order to call the routine; but it might not be in scope at the routine spec statement, and may have to be moved so that it is.]

record spec statements are not allowed in IMP80.

b) The syntax of record format statements has been extended to permit alternative formats, i.e. to enable all or part of a record to be interpreted in different ways.

Example:

```
record format RF(integer A or byteinteger B, C, D c
                 or long real E)
record (RF) R
```

The record R can be considered to consist of an integer or three byte integers or a long real. Each alternative starts at the same address. Thus it follows that in

```
record format RF2 (byteintegerarray A(0:10) or c
                   string(10) S)
record (RF2) R2
```

R2_A(i) holds the ith character of string R2_S.

Note that all the sub-fields in a record format must have distinct identifiers.

In the first example above, the three alternatives were of different sizes. This is permitted: the alternatives have padding bytes appended to them to bring them up to the size of the largest. Thus when calculating the size of a record, use the size of the largest alternative.

When only part of a record is to have alternative formats, the alternatives must be bracketed within the record format statement.

Example:

```
record format RF3(integer TYPE, real RATIO,
                  (byte integer array A(1:20) c
                   or string (10) S c
                   or record (RF2) DATA),
                  string(*) name SN)
```

More than one set of alternatives can be given within a single record format; in addition, they can be nested. Redundant brackets round alternatives are allowed.

c) Records can contain records. The format of such a record must have already been defined, or be explicit. [A record clearly cannot contain a record with the same format as it itself has.]

Records can contain multi-dimensional arrays of fixed bounds, of any type.

Records can contain record names. The format of such a record name can be the same as that of the record containing it; thus

```
record format RF4(integer X, record (RF4) name NEXT)
```

is permitted.

15. external items

a) The IMP9 keyword extrinsic is replaced by external ... spec .

Example:

```
extrinsic integer array A(1:500)        in IMP9
```

becomes

```
external integer array spec A(1:500)    in IMP80
```

External variables can be initialised, like own variables, in declaration statements, but not in specification statements.

Example:

```
external integer array A(1:500) = 25(10), 14(72),
                                  16(22), 63(*)
```

b) External variables or procedures may be given an alias. The form

```
alias "..."
```

can follow the identifier name, in declaration statements or specification statements.

Example:

```
external real function spec SIN alias "MATH$DSIN"(real A)
```

The string constant specifies the string to be used for external linkage (i.e. the external reference). From within the program the item is referred to by its identifier, in the usual way.

.6. Procedures as parameters

When a procedure has a procedure parameter the specification of the latter is given in the parameter list, not in a subsequent spec statement.

Example:

```
routine X(integer Y, routine Z(real A), string (10) S)
```

John M. Murison

Jobber

EMAS 2900 JOBBER NOTE

No: 1
Date: 15/1/79

The Scientific Jobber

The Scientific Jobber is a batch process which provides a fast, low-overhead environment for the development of ALGOL, FORTRAN and IMP programs. It incorporates a control language processor, a source program editor, a batch program loader and a set of standard mathematical procedures.

The Jobber was first developed for ICL's VME/B and VME/K operating systems, and its availability on EMAS 2900 provides a common environment for user programs across the three systems.

The speed and low overheads of the Jobber are achieved by making available a subset of the facilities provided by the full operating system, and by supporting this subset within a single program.

Input to the Jobber can be as a batch of cards assembled by the operators, or as a file of jobs sent from a user process.

Operator's view

To the operator, a Jobber process is controlled as if it were a spooled device: there is a stream called JOBBER, for which "documents" are queued until requested by a Jobber process.

The appropriate subset of Spooler commands is as follows:

S/Q JOBBER	Display the state of all Jobber processes and list the names and estimated execution times of any files of Jobber jobs waiting to be run.
S/JOBRnn?	(JOBRnn identifies a particular Jobber process). Display the state of this Jobber process.
S/ABORT JOBRnn	Stop this process; if a batch of jobs is being executed, it will be re-appended to the Jobber queue.
S/DELETEnnnnnn	Delete a file from the Jobber queue, as identified by its unique six-digit identifier.

The single Director command required is:

D/START JOBRnn	Starts a Jobber process.

User view

This is fully expounded in the ERCC 2980 User's Guide (Section F), and in the ICL 2900 series publication "Using the Scientific Jobber". In the latter, ignore references to system-dependent features such as libraries, and note that the EMAS 2900 Jobber runs in ISO mode.

There is an additional input route on EMAS 2900: the user can make up a file of jobs in a normal process and LIST or SEND it to the device ".JOBBER".

Installation manager's view

Several Jobber processes can be set up, to a maximum of one per file system. For example, the ERCC 2970 currently has three Jobbers (JOBR00, JOBR01 JOBR02), corresponding to the disc packs EMAS00, EMAS01, and EMAS02.

A Jobber process is a standard process with the following changes:

a) The basefile is set to a fixed up Jobber subsystem. This is done by the command:

```
D/SET BASEFILE JOBRnn
D/user.filename
```

b) The ACR of the process must be less privileged than MANAGR and more privileged than its users, if installation options and level 2 facilities respectively are required (e.g. ACR 5 on the ERCC 2970).

c) The maximum file size for the process must be raised, currently to 2 Megabytes; this figure may be reduced later.

When required, the process can be run with the Edinburgh Subsystem, by setting the basefile to a null string.

There exists a file in each Jobber process called JO#JOURNAL. This has READ permission granted. It contains a record of work. Specifically:

a) For each batch of jobs accessed: the name of the file and the date and time of access.

b) For each user job: the time it was begun, the CPU time used, the number of lines printed to primary output, and the number of sequential or direct access data record transfers.

When this file is full, it is listed and a new one created. It can be used as a basis for accounting outwith the standard EMAS 2900 accounting system. In the event of a major fault, this file can also be used as ultimate diagnostic output.

Installation options

A number of installation management options are available with the Jobber. There are two levels of control:

a) Options applicable to a particular batch of jobs (set by including a control card with the batch).

b) Options applicable to all jobs run by a Jobber process (set by command from the MANAGR process).

Note: Installation defaults may be set for type (a) options from MANAGR, but are overriden by any batch control card.

Options for a particular batch

The default values for the options applicable to a single batch of jobs can be modified by the insertion of a card (or line) beginning //JOBBER.

The options which may be modified are as follows:

1) Facility level

There are two facility levels for file access. These are specified by the FACLEVEL parameter, which can take the value 1 or 2 (the default). At Level 1 users are not allowed to access files and all input and output for such jobs must be via the Jobber control streams. At facility Level 2, in addition to the Level 1 facilities, creation of and access to source program and data files belonging to any process with a less privileged ACR is allowed (i.e. all normal users).

2) CPU limits

The default and maximum CPU limits for each user job can be modified, by the MDEFAULT and MMAX parameters respectively.

3) Printed output limits

The default and maximum printed output limits for each job can be modified, by the LDEFAULT and LMAX parameters, respectively.

4) File sizes

The maximum size permitted for a file created by a user can be modified (the FMAX parameter).

e.g. JOBBER(FACLEVEL=2, MMAX=20)

Options applicable to all jobs

1) The default compilation options may be modified by setting the parameter OPTIONS, which takes a value of one or more compilation options, separated by ampersands (see the ICL document "Using the Scientific Jobber").

2) The amount of printed output written to a Jobber output file can be controlled by the parameter BREAK. This parameter takes an integer value, which is a line count. Each output file will be closed and printed after the completion of the Jobber job during which this count was exceeded. A new file will be created and used for the output from subsequent jobs.

Defaults

If no action is taken to set options then Jobber will be controlled by a set of defaults, fixed in the Jobber program as follows:

```
FACLEVEL =    2

MDEFAULT =   30

MMAX     =  300

LDEFAULT = 1000

LMAX     = 5000

FMAX     = X'200000' (bytes)

OPTIONS  = ISO&QUOTES
```

Setting options from the MANAGR process

Three commands are avilable to override the default option values with which the Jobber program was compiled. (The file ERSC04.JOPTY must first be INSERTed in MANAGR's directory.)

1) To create a file in one or more Jobber processes which will contain default values to be used in preference to those used within the Jobber program (taken from file JDEFAULTS), do the following:

```
CREATE J OPTIONS (jobbers)
```

jobbers = the names of one or more Jobber processes, separated by commas

2) To change particular options, do the following:

```
SET J OPTIONS (Jobber)
```

The program will prompt for data, which should be given in the form:

DATA: KEYWORD = VALUE

The reply STOP causes a return to command level.

Keywords and permissible values are as follows:

keyword	meaning	conditions on value
FACLEVEL	Facility level	1 or 2
MDEFAULT	DEFAULT CPU time per job	10 <=MDEFAULT<=3000 sec
MMAX	Maximum CPU time per job	10 <=MMAX<= 3000 sec
LDEFAULT	Default lines output per job	50 <=LDEFAULT<=100,000
LMAX	Maximum lines output per job	50 <=LMAX<=100,000
FMAX	Maximum file size	10 <=FMAX<= 1024 Kbyte
BREAK	Output break limit	BREAK >= 100
OPTIONS	Compilation options	(See "Using the Scientific Jobber" or the ERCC 2980 User's Guide)

3) To list the current option defaults for a particular Jobber process, do the following:

PRINT JOPTIONS(jobber)

Future developments

a) Dynamic loading of external library routines as an installation option.

b) Direct-access data files.

c) Shared compilers. Currently the Jobber has its own copy of the compilers, as does the Edinburgh Subsystem. It is possible for all compilations to share a single copy of the compilers, whether executing in a Jobber process or in a standard subsystem, and in background or foreground mode.

d) Accounting. It may be possible to produce standard EMAS 2900 accounting records.

A. Anderson

Operations

EMAS 2900 OPERATIONS NOTE

No: 1
Date: 7/3/78

Messages from Oper to Executive and User Processes

Currently during an EMAS session, three executive processes are created, namely "DIRECT", "VOLUMS" and "SPOOLR".

DIRECT performs the following functions:

1. file system maintenance
2. checking of log-in messages from the network
3. creation of other processes
4. dispatch of operator messages and broadcast messages to other processes

VOLUMS performs the following functions:

1. back-up
2. archive
3. reload of files from back-up and archive, and from System 4 tapes

SPOOLR performs the following functions:

1. queueing and dispatch of files from user processes destined for local or remote output devices, or for batch execution locally or remotely
2. scheduling of batch (background) jobs for local execution
3. receipt of files for local users or for local execution from local or remote devices

These three executive processes require to handle arriving request messages "on demand", i.e. they request a message ("POFF"), go to sleep if there isn't one, and take the next one if there is. These processes are said to require their messages "synchronously", and the special message forms

D/<message>	(for DIRECT)
V/<message>	(for VOLUMS)
S/<message>	(for SPOOLR)

are designed to supply <message> to the process on a "synchronous" service number.

Messages from the Oper to other processes, however, need to operate in an essentially asynchronous manner, and are therefore sent on a separate "asynchronous" service number, which is used when an "n/<message>" is typed

(e.g. 5/ST). Unlike the System 4 implementation, however, text messages are not sent using n/<message>, but via facilities in the executive process DIRECT: SET MSG, SEND MSG, BROADCAST and MSG<username> (see below).

The special Director functions, e.g. ST, XST, are requested using n/<message>. These asynchronous messages may also be sent to the executive processes if necessary, but asynchronous messages other than the ones listed below will simply be echoed with a question-mark by any process, with no other effect.

Text messages from Oper to user processes

D/SET MSG

This causes further input to DIRECT to be concatentated into a stored string. Input is terminated by a ":" (colon) on a line by itself, or after the 255th input character. When the ":" is typed, the stored string is played back on the Oper.

D/BROADCAST

This causes the stored message (see SET MSG) to be broadcast to all processes started from interactive terminals.

D/SEND MSG <username>

This sends the stored message (see SET MSG) to the user specified (if the process was started from an interactive terminal).
[Not implemented at 7th March 78.]

D/MSG <username>

This causes the next line typed to be sent as a text message to <username>.

In the last two cases, the message will be sent to <username> even if he is not logged in, but a flag 61, "Process not available", result is given. Later versions of the Subsystem will be able to pick up stored messages at process start-up.

Special Director asynchronous messages

n/ST

This causes process n to be stopped when Director for process n is next uninhibited. This is a "safe" way of killing a process, viz files and i/o streams are properly disconnected. It should normally be "immediately" effected. The process "FAILS 14".

n/XST

This causes the process to be killed instantaneously. This is potentially dangerous, as Director procedures may currently be updating a file index, for example. This should only be used if n/ST is patently having no effect. The process "FAILS 13".

The remaining asynchronous messages are for diagnostic purposes:

n/CMP

This prints the "master page tables" on the main log.

n/DIRMON i j

This sets the Director monitor variables DIRMON, DEPTH to i, j respectively.

(0 0 means "off".)

n/SIGMON i j

This sets the Director monitor variables SIGMON, MONITORING to i, j respectively.

Bits in the SIGMON word have meanings associated with the following values:

Bit-values	Action
1	print monitor
2	print string describing contingency
4	print dump of stack
8	print dump of gla
16	print master page
32	dump code around PC
64	print regs at contingency

MONITORING=1 causes PONs, POFFs, OUTs etc from the process to be monitored on the main log.

(0 0 means "off".)

General

D/PROMPT

This causes the DIRECT process to prompt for further input. The prompt is "DIRECT:". The "D/" may then be omitted from further input, and the "COMMAND" key should not be pressed.

PROMPT 0

to DIRECT causes the "DIRECT:" prompt to be removed.

D/CLOSE

This stops DIRECT, VOLUMS and SPOOLR.

J.K. Yarwood

EMAS 2900 OPERATIONS NOTE

No: 2
Date: 5/9/79

Operation of a Card Punch Under EMAS 2900

It is normally necessary, when card punch output is being produced, for operator intervention between the punching of each file. This is because it is not always possible to tell where one punch file ends and the next begins. To alleviate the problem, EMAS 2900 uses the card punch in the following way.

Before a card output file can be punched, the operator must give the SPOOLR command

S/START CP0

This means that after each file output the card punch will stop - it will not go on to the next file. Also, at this point (i.e. after the file has been punched), SPOOLR will display a line of the following form on the OPER:

CP0 username filename size delivery

e.g. CP0 ERCC20 CARDOUTP 3K Room_2020_JCMB

The cards should then be removed from the card punch hopper, and the operator should either interpret the first card in the pack or write on the first card the information displayed on the OPER.

The cards are punched as follows:

Card 1	username filename delivery date time
Card 2	first line of file
.	
.	
.	
Card n-1	last line of file
Card n	as Card 1

Note that the date and time on the first card is the date and time at which the file was sent for punching, not the date and time it was punched.

W. Laing

No: 3
Date: 13/2/80

Automatic Issue of OPER Commands

At System start-up, and on command D/AUTOFILE at the OPER console, the DIRECT process attempts to connect a file of operator commands of the form D/<text>, V/<text> or S/<text>, each preceded by a time-of-day in the form hh.mm representing the times at which the commands are to be actioned. This Note describes the detailed operation of the scheme.

1. Default filename

At start-up the System attempts to connect the file MANAGR.AUTOFILE as the default file of commands. There is no message if the file cannot be found or is not permitted to DIRECT. Use of this default filename is inhibited by:

(a) previously destroying the file or withdrawing its permission to DIRECT, or
(b) giving command D/AUTOFILE 0 before the file system consistency check is complete, or
(c) giving command D/CLOSEUSERS before the file system consistency check is complete.

2. Assigning a file as the command file

To assign a file as the command file, or to re-assign the default or other command file, the command D/AUTOFILE is given at the OPER. The prompt "Give Auto-filename:" appears, and the next input line should be a "6.11-character" filename. For example:

```
MANAGR.AUTOFILE1
```

The effects are as follows:

* the file currently being used as a command file (if any) is disconnected
* the specified file is connected in read-mode if possible
* the relevant start-point in the file is found, as described below ("Description of operation")
* commands up to the current time are actioned

Possible responses are:

0	successful
32	file does not exist or no access permission
33	mode conflicts with a current connect mode
37	owner not found
89	invalid file

3. De-assigning the command file

The current command file is disabled by:

(a) D/AUTOFILE 0
or (b) D/AUTOFILE

which disconnects the current file and attempts to connect the file given in response to the "Give Auto-filename:" prompt.

4. File format

The file must be a character file conforming to the Edinburgh Subsystem conventions, otherwise the D/AUTOFILE command gives the result 89. Lines of the file which are to be recognised by the AUTOFILE scheme must have a time-of-day in the form hh.mm as the first five characters. All other lines are ignored. The times specified must be in the range 00.00 to 23.59; lines commencing with "times" outside this range are ignored. The times must be in ascending order, though several consecutive (non-ignored) lines may have the same time. If a line is incorrectly placed, result 89 is given by the D/AUTOFILE command. The file contents are ignored.

The rest of a line having a time as its first five characters may be:

(a) a command to DIRECT, e.g. D/USERS 30
or (b) a command to VOLUMS, e.g. V/REQLIST
or (c) a command to SPOOLR, e.g. S/BATCHSTREAMS 2
or (d) a marker line, IPL point (see "Description of operation", below)

Spaces following the time are ignored. Multiple spaces in the rest of the line are treated as single spaces. In marker lines only the characters "IPL" are significant. Commands should be in upper case. Lines not conforming to the descriptions (a) - (d) above are otherwise ignored.

Note that it is not currently feasible (for example) to initiate backup in VOLUMS, because there is no way of supplying data to the VOLUMS prompt in response to the V/BACKUP command. Methods could obviously be developed.

5. Description of operation

When a file becomes assigned as the auto-command file, either at System start-up or at a successfuL D/AUTOFILE command, the file is searched from the end to find the most recent "IPL point" marker line (if any) whose time is earlier than the current time of day. Commands from that IPL point (or beginning of file) up to lines timed not later than the current time are immediately actioned. Each command is reported in two lines output at the OPER. For example:

```
1/ Auto-file command:
1/ S/BATCHSTREAMS 1
```

Thereafter as the time-of-day reaches the time of the next command (or commands), each command is executed, with the above two-line report on the OPER. At midnight a new pass of the file commences.

6. Example of an auto-command file

```
00.00  IPL point (redundant)
00.00  D/SCARCITY 20
00.00  D/PREEMPTAT 30
00.00  S/BATCHSTREAMS 1
  :
  :
12.00  IPL point            System development
12.00  D/SCARCITY 20
12.00  D/PREEMPTAT 30
  :
14.00  S/BATCHSTREAMS 1
  :
17.00  IPL point            System development
17.00  D/SCARCITY 20
17.00  D/PREEMPTAT 30
18.00  S/BATCHSTREAMS 1
```

J.K. Yarwood

EMAS 2900 OPERATIONS NOTE

No: 4
Date: 14/04/80

Commands to OPER in SUP26C

A picture may be displayed on a screen with the command

```
P  picture  screen
```

where picture may be

```
L   OPERLOG
P   Process List
S   SPOOLR's picture
V   VOLUMS's picture
```

and "screen" is the number of the screen. The PGF and PGB keys on the keyboard apply to the interactive screen (0) only. A picture on another screen can be paged forwards and backwards with the commands

```
F   screen
```

and

```
B   screen
```

Notes

1. All OPER stations are treated equally; the concept of MAINOP disappears.

2. To display a SPOOLR picture it is necessary to do both

```
S/...
```

and

```
P  S  screen
```

Similarly for a VOLUMS picture.

A. Gibbons

EMAS 2900 OPERATIONS NOTE

No: 5
Date: 14/04/80

Commands to GPC in SUP26C

1. GPC QS dev

where dev is a device mnemonic, gives the status (i.e. NOT ALLOCATED or READY or REQUEST FIRED or SENSE FIRED or QUEUED or DISCONNECTED) of the device.

2. GPC ?

gives the status of all the GPC devices.

3. GPC CDS dev on/off

where dev is a device mnemonic and on/off is either ON or OFF, is used to connect or disconnect a device (strictly speaking a stream). A device may be disconnected from one GPC and connected on another. Only devices which are NOT ALLOCATED can be disconnected, and only devices which are DISCONNECTED can be connected.

Notes

1. When an LP is connected, the repertoire indicated by the thumbwheel settings is loaded. Thus

```
CDS OFF
CDS ON
```

can be used to change the repertoire.

2. In the case of magnetic tapes, the lowest mnemonic in a cluster (stream) should be specified. The whole cluster is disconnected and connected.

3. CDS may be used only on devices known to the system at IPL time, i.e. these given by

```
GPC ?
```

A. Gibbons

EMAS 2900 OPERATIONS NOTE

No: 6
Date: 24/4/81

CHOPSUPE and Supervisor OPER Commands

Note: Where a numeric parameter is required in the following commands, it may be specified in one of two ways: as a decimal number, e.g. 123, or as a hexadecimal number, in which case it is preceded by the letter 'X', e.g. X400.

CHOPSUPE Commands

(i) PON dsno dact p1 p2 p3 p4 p5 p6

PON a record to dsno<<16! dact with parameter 'p1' and optional parameters 'p2'.....'p6'. Unspecified parameters are not zero!

(ii) SRCE srce

Set the SRCE parameter in future PONs (see (i)) to 'srce'. (SRCE will otherwise be the relevant Oper.)

(iii) PLOT tsn file disc page npages (PLOT = Program Load Off Tape)

Load 'npages' of program 'file' from magnetic tape 'tsn' to 'disc' starting at 'page'.

E.g. PLOT MAC002 2 EMAS00 X380 256

(iv) PLOD disc1 p1 disc2 p2 npages (PLOD = Program Load Off Disc)

Copy 'npages' from 'disc1' starting at 'p1' to 'disc2' starting at 'p2'. There are no checks on this command; use with care!

E.g. PLOD EMAS00 X100 EMAS01 0 55

(v) LABEL disc vol

Write the six-character volume label 'vol' to 'disc' defined as its device mnemonic.

E.g. LABEL ED01 EMAS01

(vi) ILABEL disc vol

As LABEL (see (v)) but an IPL bootstrap program is written as part of the volume label.

(vii) FORMAT drive lc uc lt ut

Format the disc on drive 'drive' in standard EMAS format, where:

lc = lower cylinder
uc = upper cylinder
lt = lower track
ut = upper track

Each pair lc, uc and lt, ut may be replaced by -1, meaning format all cylinders or tracks respectively.

E.g. FORMAT ED21 -1 -1 formats the whole disc on ED21.

(viii) GPC text

PON 'text' to the GPC routine.

Text may be one of the following:

a) ?

Give status of all devices.

b) QS dev

Gives the status of the single device whose mnemonic is 'dev'.

c) CDS dev onoff

Where 'dev' is a mnemonic and 'onoff' is either ON or OFF. This configures the device ON or OFF. Only devices which are not allocated may be configured off, and only devices which are disconnected may be configured on.

Notes

* If the device is a tape cluster the lowest deck should be given; the whole cluster is disconnected or connected.

* LPs have their reps loaded when connected; thus CDS OFF followed by CDS ON may be used to reload the rep.

* CDS may not be used on FEPs (except when they have gone down), as these cannot yet be deallocated.

* The device may be reconfigured to another GPC between a CDS OFF and a CDS ON.

d) CDM ZXn dev

The command CDM is used to configure in to the system GPC attached devices that were not GROPED at IPL time. For each GPC having a spare stream there is a dummy device with a mnemonic of the form ZXn. To configure in say, CP0, CDM is used thus:

GPC CDM ZX0 CP0

If successful the response is: GPC: CP0 ptsm DISCNCTD

Otherwise the response is: GPC ??CDM ZX0 CP0

The device can now be made available to the system with the CDS command (see above).

Notes

* Any ZX device displayed by the command GPC ? may be used for CDM.

* CDM may also be used to configure out devices provided they are disconnected. (i.e. have been CDS'd off.)

* CDM is not available for OPERs or MT devices.

* If the required device is an FEP then the following action should be taken:

```
GPC CDM ZXn FEn
GPC CDS FEn ON
FEPUP n
```

(ix) SLOAD fsys page

Load into store and activate the supervisor which starts at 'page' on file system 'fsys'.

E.g. SLOAD 0 X40

(x) DUMP tsn disc npages

Copy 'npages' of 'disc' starting at page 0 to file 1 on magnetic tape 'tsn'.

E.g. DUMP JM0001 EMAS00 X400

(xi) PRIME tsn disc npages

Write 'npages' from file 1 of magnetic tape 'tsn' to 'disc', starting at page 0.

(xii) POFFMON n (0<=n<=1)

If 'n' is zero POFF monitoring is switched off, otherwise it is switched on.

(xiii) KMON serv n (0<=n<=1)

If 'n' is zero then switch off monitoring of kernel service 'serv', otherwise switch monitoring on.

Note that this will not take effect until the kernel is loaded.

(xiv) UNPLOT tsn file disc page npages

As PLOT (see (iii)), but from disc to magnetic tape.

(xv) INH serv

Inhibit service 'serv'.

(xvi) UNINH serv

Uninhibit service 'serv'.

(xvii) DIRVSN n

When director is to be loaded, use version 'n', where 1<=n<=3 (default is version 0).

(xviii) DT date time

Set the date and time, where 'date' is of the form ddmmyy and 'time' is of the form hhmm.

(xix) XDUMP addr len

Print a hexadecimal dump of store from virtual address 'addr' of length 'len' bytes.

(xx) REP at with

Replace the integer at virtual address 'at' with the value 'with'.

(xxi) ISR n

Display the contents of image store location 'n'. Invalid 'n' will cause a System crash!

(xxii) ISW n x

Replace the contents of image store location 'n' with 'x'.

(xxiii) SHOW addr len

Display 'len' bytes of data from address 'addr' (len <= 64).

Supervisor commands

(i) The following CHOPSUPE commands are also available in Supervisor:

PON	SRCE	PLOD	PLOT	GPC	DUMP	PRIME
KMON	UNPLOT	INH	UNINH	DIRUSN	XDUMP	REP
ISR	ISW	SHOW				

(ii) STARTD

Restart DIRECT process after it has failed or been accidentally stopped.

(iii) OPER <text>

Will pass text to the Oper handler; not yet implemented.

(iv) DDUMP label page

Read 'page' from the disc with label 'label' and dump it on the Mainlog in hex.

E.g. DDUMP BUSH20 X800

(v) SLAVES n

Turn all slaves off (n=0) or on (n=-1).

(vi) P ident n

Display picture 'ident' on screen 'n' of the Oper on which the command was entered.

Idents are currently as follows:

P = Process list
L = Oper Log
S = Spooler Picture

'n' (0<=n<=3) gives the screen no, 0 being the interactive screen. If 'n' is omitted 0 is assumed.

(vii) B n (1<=n<=3)
F n

Page the picture currently on screen (displayed by the previous command) Backwards or Forwards by one screenful. Screen 0 is manipulated by the Page Forward and Page Backward keys.

(viii) TRACE

This command is provided to control tracing options in special tracing Supervisors used in research. It has no effect on standard Supervisors.

Dual Supervisors support also:

(ix) OCP n onoff

(where onoff is either ON or OFF) Configures OCP n ON or OFF.

(x) SAC n onoff

Configures SAC n ON or OFF.

(xi) SMAC n onoff

Configures SMAC n ON or OFF.

J. Maddock
P.D. Stephens

—Preliminary—

EMAS 2900: Summary of facilities for controlling interactive access

Control is available to the System Manager under the following headings:

i limitation of maximum numbers of interactive users, both globally and in subgroups based on user identifiers. Process 1, DIRECT, provides the control via the D/USERS command, which has been described in a separate document.

ii limitation of interactive session length (clock time). A default of infinity applies unless the D/SESSION LENGTH command is used to reduce it. In addition the maximum session length can be set for individual users by means of the MANAGR command SET SESSION LENGTH. The default or individual setting may be in the range 5 minutes to 4 hours.

iii limitation of OCP time per interactive session. This is not currently implemented, but would follow the pattern of ii above, with a global default, re-settable, and the ability to set specific maxima for individual users.

iv limitation of OCP time per interaction. This also is not currently implemented, and might be based on limitations on calls on DSET IC, relating same to calls on REQUEST INPUT or REQUEST OUTPUT. Satisfactory means of implementation have not yet been proposed.

v limitation of access to the system at times at which resource for interactive computing is deemed to be scarce. A scheme is proposed in a paper (17.11.79) by RR McLeod for the Edinburgh Charging and Allocation Committee. In summary, each user accreditted to the system has a share in the scarce resources; the share units are distributed regularly, and are consumed only during times of scarcity, currently defined simply by a level of interactive use (number of interactive users). Users who currently have no remaining scarce-resource units are denied access to the system at times of scarcity. The accompanying paper gives full details of the implementation.

It is believed that i, ii and v above will form a satisfactory and sufficiently flexible control scheme for the three currently - envisaged installations.

K. Yarwood
5th December 1979

— Preliminary —

EMAS 2900: Director Implementation details for the scarce-resource units scheme

1. Allocation of Units

The Custodians of the scarce-resource units hold a "share-register", being a specification of what proportions of units are to be distributed to faculties, departments, groups of users or individual users. Director procedure DSFI is called, by a privileged program operated by the Custodians, to distribute units according to the register. The DSFI procedure can also be used by a privileged program to assign an individual user to a group-holder of scarce-resource units. The effects of this assignation are as follows:

i when an assigned user is eligible to have his units decremented (viz. during a period of scarcity), units are actually subtracted from his group-holder's total. (The assigned user has no units of his own: the distribution program must ensure that "his" units are given to his group-holder).

ii when an assigned user reads his current unit total, using procedure DSFI (non-privileged entry), the value obtained is the total owned by his group-holder.

In addition, any user (not being an assigned user) may give some or all of his units to any other user using the non-privileged Director procedure DDONATE.

The following details will appear in the EMAS 2900 Subsystem Writer's Manual in due course:

i Extensions to Director procedure DSFI

Type = 33	Get or Set scarcity ration. An integer value is returned or set, respectively being the number of scarce-resource units. The units are notionally 0.01 pence as derived from the installation charging formula. In the case of Get (but not Set) scarcity ration, if the user is assigned to a group-holder of units the value relates to the group-holder's ration. The "Set" option is available only to privileged calling programs.
Type = 37	Get or Set a group-holder to whom USER is (or is to be) assigned. The parameter value is a string (6) <username>. The "Set" option is available only to privileged calling programs. If a USER is already assigned to a group-holder when the "Set" option is called, the old group-holder username is discarded. No user should be assigned to a group-holder username which itself is assigned to a further group-holder username; however, Director makes no check on/

on the data-structure being created, and relies on the Custodians' programs to keep the structure valid.

ii New Director procedure DDONATE

external integerfn DDONATE (string(6) USER, int FSYS, UNITS)

This procedure enables the process owner (provided he is not assigned to a group-holder) to give some or all of his current scarce-resource unit total to any other USER (on disc FSYS) on the system. If UNITS exceeds the process owner's current total then it is limited to his current total. If UNITS is negative the procedure call has no effect. Otherwise UNITS are subtracted from his current total and added to USER's total, or to USER's group-holder's total if USER is assigned to a group holder.

Possible error results: 37, 60.

iii New fields in the user-information-segment, record format UINFF (available on the 2970 in file DIRARC.CFILE_UINFF)

SCARCITY	the number of concurrent interactive processes at or above which interactive computing resource is deemed to be scarce.
PREEMPT AT	the number of concurrent interactive processes at or above which users having no scarce-resource units are liable for pre-emption (i.e. automatic logoff) as a result of users logging on who do possess units.
SESSEND	(this field is not related to the subject of this document!). The session-end time (as seconds from midnight) at which the process will be automatically terminated. Zero indicates that there is no specified automatic session-end outstanding for the process. This field is set only at process start-up, according to the default or individual maximum session length setting for the process. (It will not be reset if a process becomes eligible for pre-emption during a time of scarcity, nor will it be reset if a system-close time is specified by the machine operators after the process has started). SESSEND will be set to < session length > past midnight if the period from process start-up to session end spans midnight.

RESUNITS/

RESUNITS	this field is always a copy of the process owner's current scarce-resource units total (or that of his group-holder if he is assigned to one).

2. Consumption of Units

A number of concurrent interactive users (by default being "large") is taken to define a level at and above which interactive computing resource is deemed to be scarce. At and above this level, each current user's scarce-resource units are decremented (but not below zero) according to the installation charging formula. This applies both to batch and interactive sessions. The number of interactive users defining scarcity may be re-set by the DIRECT command

D/SCARCITY n

3. Control of Access to the System

Interactive resources are deemed to be scarce if the number of interactive processes equals or exceeds a value, n, "large" by default and re-settable by the DIRECT command D/SCARCITY n.

When a user attempts to log on, following the checks against restrictions of users belonging to specified groups the scarce-resource tests are performed as follows:

i If resources are not currently scarce, log-on is allowed whatever the user's (or his group-holder's) scarce-resource unit total. However, if his total is currently zero, his process is marked for possible pre-emption as described below.

ii If resources are currently scarce and the user has no scarce-resource units, the log-on is rejected with the message "NO RESOURCE".

iii If resources are currently scarce and the user has scarce-resource units, the log-on is allowed. In addition, if the current number of interactive processes exceeds a value, n, "large" by default and re-settable by the DIRECT command D/PRE EMPT AT n, then the oldest process if any is marked for pre-emption as described in i above, and that process is terminated in about 6 to 8 minutes, with 5 and 2-minute "END OF SESSION" warnings.

Initiation of batch jobs will be subject to the following additional controls (implemented by SPOOLR):

i no jobs with priority less than HIGH will be initiated at times of scarcity.

ii no jobs will be run at times of scarcity for users whose scarce-resource units (or those of his group-holder) are zero.

K. Yarwood
5th December, 1979

Service

EDINBURGH REGIONAL COMPUTING CENTRE

EMAS 2900 SERVICE NOTE

No: 1 (version 5)
Date: April 1980

Conversion to EMAS 2900 for 4-75 Users

N.B. THIS NOTE REPLACES SERVICE NOTE 1 DATED AUGUST 1979.

Introduction

This note is intended for existing users of EMAS on the 4-75 who are planning to move to EMAS 2900. It primarily concerns differences in the appearance of the Subsystem, but other areas are mentioned where relevant. It reflects the state of the System as running on the ERCC 2970 at the time of writing. More up-to-date information can be found in the ALERT and HELP information.

Essential background information

It is assumed that the reader has access to

* EMAS 2900 User's Guide, Second Edition, January 1980
* EMAS 2900 Information Card, Third Edition, February 1980
* For IMP users, the IMP Language Manual, and particularly its Update 1 (February 1978)

Design intention

The user interface has been based on that used for EMAS on the 4-75. The re-implementation for ICL 2900 Series computers has been used as an opportunity to tidy up some of the inconsistencies in the command language, in the light of experience of the problems which users have encountered on the 4-75. The intention is that it should be possible to carry out all the same operations - often using identical commands.

The rest of this note should be read in conjuction with the EMAS 2900 User's Guide.

Chapter 1

File names are constructed as on the 4-75 except that both file names and partitioned file member names can be up to 11 characters.

Chapter 2

The device code .TT is no longer valid. In its place the more general forms .IN and .OUT must be used. They refer respectively to the current primary input and output. For foreground sessions these are normally both the interactive terminal. Hence:

IMP(A,AY,.OUT)

Use HELP(DEVICES) to obtain an up-to-date list of output devices.

Chapter 3

When logging on, respond to the "HOST" prompt with "2970" or "2980", not with "EMAS".

Chapter 4

Note the distinction between SS# and T# files. Note also that the name of the default compiler listing file is T#LIST.

Chapter 6

The command FILEANAL is replaced by ANALYSE, which takes three parameters:

ANALYSE (file, option, out)

See page 6-1 for details.

COPYFILE becomes COPY.

Chapter 7

The distinction between Character files and Data files is more significant on the 2900. In particular the text editors only operate on Character files. The provision of a format "C" in DEFINE makes it possible to generate Character files as output from a FORTRAN program.

The first parameter for DEFINE is now an integer; hence:

DEFINE(3,FILEAB) not DEFINE(FT3,FILEAB)

Note that DDLIST is replaced by DEFINE(?).

Chapter 11

The program loading mechanism is significantly different. However, the majority of users will not be greatly affected.

INSERT replaces INSERTFILE. It must be called explicitly for object files, other than main programs, before they are first used.

The function of APPENDLIB is now replaced by a facility provided by the OPTION command. Briefly:

APPENDLIB (CONLIB.GRAPHICS) becomes

OPTION (SEARCHDIR=CONLIB.GRAPHICS)

There is a new command ALIAS which makes it easy to modify the names of commands.

Chapter 12

The first 31 characters of entry names are significant.

TESTINT is replaced by INTERRUPT.

SSINFO is replaced by two functions UINFI and UINFS.

Chapter 16

The implementation of BACKSPACE is incomplete. In the case of Character files there is no restriction. In the case of Data files consecutive calls of BACKSPACE are faulted.

Chapter 17

The OPTION command provides more facilities than its counterpart on the 4-75. In particular it replaces USERLIB, APPENDLIB and REMOVELIB. See page 17-5.

Command changes

The list on the next page gives the names of 4-75 EMAS commands, their EMAS 2900 equivalents (where different) and an indication of which commands are significantly different on EMAS 2900.

EMAS 4-75 command	EMAS 2900 equivalent (if different)	Significant differences	Notes
ACCEPT			
ALERT			
ALGOL			
APPENDLIB	OPTION	*	See Chapters 11 and 17
ARCHIVE			
CHERISH			
CLEAR			
CONCAT			
COPYFILE	COPY		
CPULIMIT			
DDLIST	DEFINE(?)		
DEFINE		*	
DEFINEMT			
DELETEJOB	DELETEDOC		
DELIVER			
DESTROY			
DETACH		*	See Chapter 16
DISCARD			
DISCONNECT			
EDIT			
FILEANAL	ANALYSE	*	
FILES			
FINDJOB	DOCUMENTS		
FORTE			
HAZARD			
HELP			
IMP			Changes in language
INSERTFILE	INSERT		
LIBANAL	ANALYSE	*	
LINK			
LIST			
LOOK			
METER			
NEWPDFILE			
NEWSMFILE			
OBEYFILE	OBEY		
OFFER			
OPTION		*	See Chapter 17
PARM			
PASSWORD			
PERMITFILE	PERMIT		
PERMITLIB	PERMIT	*	No direct equivalent
PROJECT			Not available
QUEUES	DOCUMENTS		
RECALL			
REMOVEFILE	REMOVE		
REMOVELIB	OPTION		See Chapters 11 and 17
RENAME			
RESTORE			
RUN			
SEND			
STOP			
SUGGESTION			
TIDYLIB	TIDYDIR		
USERLIB	OPTION	*	See Chapters 11 and 17, and command NEWDIRECTORY
USERS			

R.R. McLeod

Transferring Files between the 2970 and 2980: T080 and T070

The commands T080 and T070 are available for transferring a user file from one mainframe to another. If it is a character file it is dispatched immediately along communications lines to the target machine. Otherwise it is sent along communications lines as a background job, so that in due course a line printer report of the execution of the job (on the target machine) will be received. There is no restriction on the type of file which may be transferred, but the size of the file cannot be greater than 1023 Kbytes.

Establish access to the commands as follows:

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)      (2970)

Command:OPTION(SEARCHDIR=ERCLIB.GENERAL)      (2980)
```

There are two forms of the command: T070 and T080, where the suffix indicates the target mainframe. The parameters for both commands are as follows:

```
Command:TOxx(file, newowner.newfile, back, overwrite)
```

file	-	the file or partitioned file member to be transferred (may belong to another user so long as read access is permitted).
newowner.newfile	-	if newowner is omitted the owner of the process issuing the command is used. If newfile is also omitted the file is given the same name on the target machine.
back	-	the background password of the receiving owner on the target machine.
overwrite	-	OVERWRITE may optionally be specified: if file "newfile" already exists in the target machine process, it will then be overwritten.

Please contact your local Advisory Service if any difficulties are encountered in using the above facilities.

A. Shaw

Accounting and Charging Routines

These routines are held in object file ACCNTS.PDCHARGE_CHARGEY on the 2970 and 2980; the corresponding source file is ACCNTS.PDCHARGE_CHARGEnS (2970 only).

The routines are described below.

externalroutine CHARGES (test)

This routine takes an optional parameter, either a specific username or the word TEST; in either case a test run is implied. The effects are described below.

If no parameter is given, CHARGES reads and clears the file index accounting information for every accredited user in the System. The information read is put into a store map file called JUSE which is then sent to the JOURNL process via the SPOOLR JOURNL queue.

Before CHARGES does this, however, it makes use of the file created for JOURNL to update a weekly accounting file and a "cumulative use" file.

The current accounting week is established and a file for the week's accounting is created unless it already exists. The form of the file name is A29nnmmmn, e.g., A2980JUN2.

The format of the file is as follows:

(1) "ddmmyyddmmyy"

the dates (inclusive) of the accounting week

(2) #n dd/mm/yy

n	specifies the charge code 1, 2, 3, 4 .....
dd/mm/yy	gives the date on which the charges following were incurred

(3) hh.mm.ss AAAAnn ppp

hh.mm.ss	is the time of day at which the charge was calculated
AAAAnn	is the relevant user number
ppp	is the charge (in pence) incurred

Line (1) appears once only, at the start of the file. Following each occurrence of a line with format (2) there are a number of lines with format (3). The charge code determines what sort of charge is being made in the lines following:

1	use of 2980 or 2972
2	file space on 2980 or 2972
3	use of 2970
4	file space on 2970

On each call of the CHARGES routine (with the exception of test runs), two sets of charges, use and file space, are calculated for the relevant machine and appended to the current accounting week file. This file is then normally accessed during the following accounting week (i.e. when it is complete).

The "cumulative use" file is called CUSE. It is created by CHARGES if necessary. It can be listed, in whole or in part, by use of the command LISTCUSE, described below.

When CHARGES is called with a parameter, a test run is carried out. In this case the file indexes are examined but not cleared, the weekly charge filename is not prefixed by A29nn, and the names of the files generated (JUSE, CUSE, MMMn) all have "TEST" appended. JUSETEST is not transferred to the JOURNL process; it will be overwritten in any subsequent test run.

If the parameter is a user name then only the charges for that user are calculated. If the parameter is TEST then the charges for all accredited users are calculated.

Notes

* If file JUSE is found by CHARGES to exist, CHARGES starts by sending it to the JOURNL queue. This should not happen, since CHARGES should have done this the previous time it was called.

* The structure of the JUSE file is described below.

* The structure of the "cumulative use" file is described below.

* At the time of writing, CHARGES is invoked nightly by a batch job which causes itself to be run the next day also. Thus the charging scheme is automatic and no operator intervention is required.

* The processing of the completed weekly accounting files is carried out as a separate stage.

---

externalroutine LISTCUSE (string(255) S)

This routine either creates and initialises the "cumulative use" file added to by CHARGES (see above), or, if the file exists, lists the contents.

The form of a call of the routine is as follows:

```
Command:LISTCUSE(usermask,out,file)
```

All three parameters are optional.

usermask — can be null (the default), meaning that information on all users is to be output, or
a six character username, meaning that information on that user only is to be output, or
a name including one or more '?' signs, specifying a group of users about whom information is to be output; e.g. ERCCI?? would give information on all users whose names started ERCI.

out	can be null (the default), meaning that output is to be sent to the primary output device (normally the user's terminal), or an output device (e.g. .LP), or a filename.
file	by default this is CUSE. If a test CUSE file is to be examined (see above), this parameter must be specified as CUSETEST.

If usermask is non-null and does not contain six characters, an error is flagged.

```
externalroutine LISTJUSE (string(255) S)
```

This routine lists the complete contents of the store map file JUSE (or JUSETEST). The form of a call of the routine is as follows:

```
Command:LISTJUSE(file,out)

file  is JUSE by default

out   is .OUT by default
```

Refer to the description of DSFI in the EMAS 2900 Subsystem Writer's Manual for explanations of the various items listed.

```
externalroutine GETUSE(string (6) USER, integer FSYS, RESET, ADR)
```

This routine returns accounting information in a record array starting at ADR, for all accredited EMAS users (if FSYS is -1), or for all users on the disc pack specified by FSYS, or for a single user (if USER is non-null). In the latter case FSYS can be used to specify where the user's file index resides, if known.

The record array starting at ADR is assumed to be (0:n) (STATSF):

```
recordformat STATSF(string(6) USER, c
   integer NEXT, FSYS, KINST, PTURN, KBTSLDEV, KBFSLDEV, c
   MSOCP, CONNECT, AFILE, AKB, DFILE, DKB, CFILE, CKB)
```

The records are held in a list structure to give alphabetic order (_NEXT giving the next item in the list). This list is terminated by a dummy record held in array element 0, with (0)_USER="ZZZZZZ". The genuine last user record thus has _NEXT=0.

The 0th record, on return from GETUSE, holds two items of information:

(0)_NEXT gives the FIRST record in the list structre
(0)_FSYS gives the TOTAL number of (genuine) user records in the structure.

If on entry (0)_NEXT is non-zero, this indicates that a record array structure already exists. In this case each user's accounting information is added in to the corresponding record (unless no such record currently exists, when on is created). When (0)_NEXT is non-zero on entry, it is assumed to point to th start of the structure, and (0)_FSYS is assumed to give the total number of user records in the structure.

If all the file systems are searched and more thasn one file index is found for a particular user, then the accounting information for each is stored in a separate record of the record array.

If RESET is 0 on entry, the accounting information is merely read from the user's file indexes. If it is 1, the file indexes are also reset.

It is expected thet GETUSE will be called on a regular basis with RESET=1, and that either all the discpacks will be searched in a single call, or that GETUSE will be called for each discpack in turn, the calling program having used GET AV FSYS beforehand.

The user of GETUSE must be running at an ACR level of 5 or less.

Structure of JOURNL File JUSE[TEST]

This file has a standard Edinburgh Subsystem file header. It is a store map file.

Following the file header, the first two words contain the packed date-and-time and the pattern X'FFFFFF03' respectively.

The remainder of the file is used to hold a record array of the structure described in GETUSE (above).

Structure of Summary File CUSE[TEST]

This file has a standard Edinburgh Subsystem header. It is a store map file.

Following the header, there are two 8-byte strings (each allocated ten bytes) which give the start and finish dates corresponding to the information held.

Thereafter a record array is mapped onto the file, from byte 20 after the end of the Subsystem header. The record array is defined as folows:

```
recordformat CRECF(string(6) USER, integer NEXT, USEP, FILEP)

recordarray CREC(0:1024) (CRECF)
```

where USER is a user name NEXT points to the next user (in alphabetical order) USER is the accumulated use charge in pence for that user FILEP is the accumulated file charge in pence for that user.

Note that:

```
CREC(0)_USER = "ZZZZZZ"
CREC(0)_NEXT = array element containing first genuine user
               alphabetical order)
CREC(0)_USEP = total number of genuine users in array.
```

The 0th element of the array is used to terminate the list. Thus CREC(i)_NEXT=0 if the user in element 1 is the last one in the alphabetical list.

J.M. Murison

EMAS 2900: Accounting and Charging Routines

These routines are held in the object file ERCI05.CHARGE (the corresponding source file is ERCI05.CHARGES), and are as follows:

externalroutine CHARGES

This routine takes no parameters. It reads and clears the file index accounting information for every accredited user in the System. The information read is put into a store map file (currently called JUSE) which is dispatched to the JOURNL process via the SPOOLR JOURNL queue.

Before CHARGES does this, however, it makes use of the file created for JOURNL in 3 ways:

1) A character file (currently called AddmmU, where ddmm is part of the date of the run of the program) is created and dispatched to System 4 EMAS process ACCNT4. The format of each line of this file is:

   hh.mm.ss AAAAnn ppp

   where hh.mm.ss is the time of day
   AAAAnn is a user number
   ppp is the charge in pence attributed to the user because of his use of the machine (hence the U at the end of the filename).

   The contents of the JOURNL file are used to calculate the charges, the current charging formula being used. This charging formula is implemented in integerfn USECH, which is local to CHARGES, and an explanation of the constants involved is given there in comment statements.

2) A character file (currently called AddmmF) is created and dispatched to System 4 EMAS process ACCNT4. The format of each line of the file is identical to that for the previous file. The charge (ppp) is that attributed to the user because of the file space currently belonging to him (hence the F at the end of the filename).

   The contents of the JOURNL file are used to calculate the charges, the current file space rates being used. These rates are implemented in integerfn FILECH, which is local to CHARGES.

3) A store map file (currently called CUSE) is updated if it exists at the time CHARGES is invoked. This is a "cumulative use" file whose contents can be output, in whole or in part, by use of the command LISTCUSE, described below.

Notes

* If a JOURNL file is found by CHARGES to exist, CHARGES starts by sending it to the JOURNL queue. This should not happen, since CHARGES should have done this the previous time it was called.

* If either of the character files of charging information is found by CHARGES to exist, then CHARGES appends the charges calculated in its current call to these files. Again, this should not happen.

* The structure of the JOURNL file is described below.

* The structure of the "cumulative use" file is described below.

---

externalroutine LISTCUSE (string(255) S)

This routine either creates and initialises the "cumulative use" file added to by CHARGES (see above), or, if the file exists, lists the contents. In the latter case it also gives the user the option to re-initialise the file.

The form of a call of the routine is as follows:

```
COMMAND:LISTCUSE(usermask,out)
```

Both parameters are optional.

usermask — can be null (the default), meaning that information on all users is to be output
or a six character username, meaning that information on that user only is to be output
or a name including one or more '?' signs, specifying a group of users about whom information is to be output; e.g. ERCCI?? would give information on all users whose names started ERCI.

out — can be null (the default), meaning that output is to be sent to the primary output device (normally the user's terminal)
or an output device (e.g. .LP)
or a filename.

If usermask is non-null and does not contain six characters, an error is flagged.

In all cases however the user is prompted:

```
Reset CUSE?
```

to which he should reply

Y <CR> if the file is to be cleared

or N <CR> or just <CR> if it is not to be cleared.

```
externalroutine GETUSE(string (6) USER, integer FSYS, RESET, ADR)
```

This routine returns accounting information in a record array starting at ADR, for all accredited EMAS users (if FSYS is -1), or for all users on the disc pack specified by FSYS, or for a single user (if USER is non-null). In the latter case FSYS can be used to specify where the user's file index resides, if known.

The record array starting at ADR is assumed to be (0:n) (STATSF):

```
recordformat STATSF(string(6) USER, c
    integer NEXT, FSYS, KINST, PTURN, KBTSLDEV, KBFSLDEV, c
    MSOCP, CONNECT, AFILE, AKB, DFILE, DKB, CFILE, CKB)
```

The records are held in a list structure to give alphabetic order (_NEXT giving the next item in the list). This list is terminated by a dummy record held in array element 0, with (0)_USER="ZZZZZZ". The genuine last user record thus has _NEXT=0.

The 0th record, on return from GETUSE, holds two items of information:

(0)_NEXT gives the FIRST record in the list structure
(0)_FSYS gives the TOTAL number of (genuine) user records in the structure

If on entry (0)_NEXT is non-zero, this indicates that a record array structure already exists. In this case each user's accounting information is added in to the corresponding record (unless no such record currently exists, when one is created). When (0)_NEXT is non-zero on entry, it is assumed to point to the start of the structure, and (0)_FSYS is assumed to give the total number of user records in the structure.

If all the file systems are searched and more than one file index is found for a particular user, then the accounting information for each is stored in a separate record of the record array.

If RESET is 0 on entry, the accounting information is merely read from the users' file indexes. If it is 1, the file indexes are also reset.

It is expected that GETUSE will be called on a regular basis with RESET=1, and that either all the discpacks will be searched in a single call, or that GETUSE will be called for each discpack in turn, the calling program having used GET AV FSYS beforehand.

The user of GETUSE must be running at an ACR level of 5 or less.

Structure of JOURNL File JUSE

This file has a standard Edinburgh Subsystem file header. It is a store map file.

Following the file header, there is a standard identification string. This is inserted by use of externalroutine JHEAD, which is passed the address of the start of the user data and a code string indicating the use of the file. JHEAD inserts the identification string in the file, and updates the address to point to the next free space in the file.

The remainder of the file is used to hold a record array of the structure described in GETUSE (above).

Structure of Summary File CUSE

This file has a standard Edinburgh Subsystem header. It is a store map file.

Following the header, there are two 8-byte strings (each allocated ten bytes) which give the start and finish dates corresponding to the information held.

Thereafter a record array is mapped onto the file, from byte 20 after the end of the Subsystem header. The record array is defined as follows:

```
recordformat CRECF(string(6) USER, integer NEXT, USEP, FILEP)

recordarray CREC(0:1024) (CRECF)
```

where USER is a user name
NEXT points to the next user (in alphabetical order)
USER is the accumulated use charge in pence for that user
FILEP is the accumulated file charge in pence for that user.

Note that:

CREC(0)_USER = "ZZZZZZ"
CREC(0)_NEXT = array element containing first genuine user (alphabetical order)
CREC(0)_USEP = total number of genuine users in array.

The 0th element of the array is used to terminate the list. Thus CREC(i)_NEXT=0 if the user in element i is the last one in the alphabetical list.

J.M. Hurison

Spooler

SPOOLER PROCESS 2900

No: 0
Date: 25/4/77

USING THE SPOOLER

CARDS FROM INPUT SHOULD BE ENCLOSED BY TWO CARDS

1) //FILE␣ (USERNAME, FILENAME)

2) //

DIRUSN 2 NOMINATE DIRECTOR ELTE X280 ON SYSTEM DISK

PON 3 1 4 START PROCESS 4 (THE SPOOLER)

SHOULD REPLY 84/SPOOLER USN STARTS

4/STOP STOPS THE PROCESS

AVAILABLE COMMANDS

4/PRIME NN8888 (COPY FILES ON SPECIFIED TAPE INTO FILE SYSTEM)

TAPE SHOULD BE WRITTEN BY MAC'S PROGRAM VTAPE ON EMAS.

ANY FAULTS AND OUTPUT TO Bill Laing 30/3/77

4/RELOAD NN8888 n m (Copy chapters specified from tape
or 4/RELOAD NN8888 n to file system. Tape should be written as above.)

4/LI (USERNAME, FILENAME) Causes the line printer / supervisor would write specified file to the p When finished returns the printer to supervisor.

Bill Laing 25/4/77

UP TO 4 PRIMES OR RELOADS CAN BE DONE ~~SIMULTANEOUSLY~~ AT ONE TIME. AT PRESENT IF A FILE EXISTS IN THE FILE SYSTEM IT IS DESTROYED AND THE NEW ONE ON TAPE PRIMED.

ERRORS

1) BAD PARAMS
2) TOO MANY REQS (More than 4 at once.)
3) READ PAGE FAILS (A page could not be transferred from tape; the file is not reloaded)
4) POSITION TAPE FAILS (Skip to tape mark failed.)

4/ALLOC CRD The card reader is allocated to the spooler and either cards are read or a status message is output.

4/DEALLOC CRD This deallocates the card reader and stops reading.

No: 1
Date: 27/3/78

Design Aims and Philosophy

These notes are a description of the design aims and philosophy behind the Spooler process (SPOOLR) for EMAS 2900. They are intended as discussion notes for the EMAS 2900 team although they may be used to gain an overall impression of the facilities which will be provided by SPOOLR. Many points may change in detail but few should change in principle.

Contents

0. Introduction

1. Definitions

2. Structure

3. Input of Documents

4. Output of Documents

5. Controlling Spooler

6. Spooler Configuration

7. Document Scheduling Parameters

8. Document Control Commands

9. Operator Commands

Relevant Papers

Director: Starting Batch Jobs

Communications: EMAS 2900 Communications Design and Implementation Notes

0. Introduction

The Spooler executive (SPOOLR) is a privileged System paged process which is automatically started by the System after an IPL. SPOOLR maintains a File Index on each of the on-line file system devices. In these Indexes it keeps queues of batch jobs to be run, and queues of files to be output to local peripherals and remote job entry (RJE) stations.

The facilities provided by Spooler include the control of local slow peripherals and communication with the Front End system to handle remote traffic. It accepts card, paper tape and RJE input to the System and carries out the spooling of all output to line printers, card punches, paper tape punches and RJE stations. It also controls the queuing of batch jobs, either detached from interactive terminals or entered on cards or from RJE stations.

1. Definitions

Document

A unit of spooling comprising control information and a file. Each document has a unique identifier. All references to documents are via their identifiers.
Note that the Spooler is not interested in the contents of the file, only in the control information.

Document Scheduling Parameters

The items in the control information which is part of each document. This information describes the document, e.g. its size, where it is going, where it came from.

File

In the context of the Spooler's work, various types of file, and documents without control information, are identified; e.g.:

* A job file, containing a complete batch job to be processed.

* A listing file, for example to be sent to a line printer.

* A file of binary data, for example containing graph plotter output.

Queue

A set of documents, ordered by priority, and size or time within priority levels.
Queues are served by streams (see next definition).

Streams

A communications path to a specific destination or from a specific source. Four types of stream are identified:

* Streams to local and remote peripherals (i.e. output communications streams)

* Streams from local and remote peripherals (i.e. input communications sreams)

* Batch job streams

* Streams to processes which receive queued files, e.g. JOBBER

Documents are received by SPOOLR on input streams and are put into queues, as determined by their scheduling parameters. Each such queue (e.g. a line printer queue) can be serviced by several streams (e.g. a stream for line printer 0, a stream for line printer 1, etc.). Whenever a queue contains a document appropriate to a stream currently inactive, SPOOLR activates the stream to deal with the document.
Streams are "queue servers".

Document Control Language

This language enables control to be exercised over documents after they have been queued.
Available to: Mainframe Operators
Remote Operators
Users

Each of these groups has different access.

Spooler Configuration

This is a file set up by the System Manager to define queues, streams, their attributes, the connections between them, etc. It is read by Spooler at each IPL to construct its tables and queues.

Operator Commands

These are commands available to local and remote operators for controlling streams.

2. Structure

A document and its descriptor, which is constructed from the document scheduling parameters, is held in the file system from which it originated.

Each queue is a linked list of document descriptors. After a System crash the queues can be completely rebuilt from the document descriptors.

When a document is placed in a queue all of the streams which serve this queue are searched, and if one is idle it is kicked to start the stream to serve the queue. If none is idle the document is just queued.

At Spooler start up, depending on the default status of the streams (either open or closed), the local streams are kicked or left idle. If the stream is to or from a remote device it is idle until the RJE terminal logs on, and is accepted or rejected depending on the status of the stream (either open or closed).

Batch streams are idle until process 1 (DIRECT) requests a number of batch streams to be started.

Streams which give documents to other processes, e.g. Jobber, are idle until such a process starts and requests a document.

Streams have default attachments to queues, but these may be changed dynamically. For example, if the local line printer is down it may be useful to serve the 'LP' queue with a remote line printer stream. When a document enters the Spooling system it acquires a unique identifier. This identifier consists of the file system number on which it is queued and a number in the range 1 to the number of descriptors held per file system. All subsequent references to the document are made using this unique identifier, which never changes.

All data transfers take place directly between files and a) devices on communications streams, b) batch jobs, or c) file processors (e.g. Jobber).

3. Input of Documents

Documents are input in two ways to the Spooler:

1) From other processes running in the EMAS 2900 System. These input documents are sent by users to the Spooler, either to be queued for batch execution, queued for printing or queued to be sent to an RJE.

2) From input communications streams. These documents come from local card readers, local paper tape readers, etc, or from RJE stations. They are dealt with according to their control parameters. They may be batch jobs for execution, files to be read into the file system, or files to be queued for printing or for output on another stream.

4. Output of Documents

Documents for output or execution are queued before being served by a stream. Queues are served by streams of three types:

1) Send File (Device type)	Sends a file to a local or remote device.
2) Start Process (Job type)	Initiates a batch job to process the document.
3) Give to Process (Process type)	Sends a file to a process which requests documents.

These queue servers only take files from queues when the server's parameters match those of the document to be handled. For example, the cpu time required by a job must be less than or equal to the current maximum on that stream; similarly with priority.

The 'Device Type' stream interface (type 1)

This uses the Communications Controller commands, i.e. CONNECT STREAM, ENABLE STREAM, etc. See the document "EMAS 2900 Communications Design and Implementation Notes".

The 'Job type' stream interface (type 2)

Batch input

The message to start a batch job is of the following structure:

```
integer DEST, SRCE, IDENT, FSYS, string (11) SPOOLR FILE
```

Contents of SPOOLR FILE:

```
VSN          integer
FSYS         integer
USER         string (6)
BACKPASS     string (6)
IDENT        integer
INST CNTR    integer
FILENAME     string (31)
OUTDEVICE    string (15)
```

The reply is of the form:

integer DEST, SRCE, IDENT, FLAG

FLAG:
1 - SYSTEM FULL
2 - INVALID USERNAME
3 - INVALID PASSWORD
4 - ALREADY LOGGED ON
5 - CANNOT START PROCESS
6 - WORK FILE FAILURE
7 - NO USER SERVICE
8 - SPOOLR FILE NOT AVAILABLE

When a batch job stops, a message is sent to Spooler from process 1 (DIRECT):

recordformat DEST, SRCE, integer ID, CONCURR, MOREJOBS, REASON

CONCURR = Number of processes still running with username.
MOREJOBS = Number of batch process to be started.

Note that if ID=-1, only MOREJOBS is relevant.

The 'Process type' stream interface (type 3)

a) REQUEST — Process asks Spooler for next file in specified queue. Spooler replies when it has a file to send:

REPLY name of file

b) FINISH — Process replies that it is finished with file.

c) REQUEUE — Process requests spooler to requeue file.

5. Controlling Spooler

There are four ways of controlling the Spooler process. These are more fully described in sections 6 to 9, but a brief explanation is given here.

1) Configuring Spooler

Each time the Spooler is started, a configuration is read in which defines the queues and streams to be used. This configuration is normally defined by each installation's System Manager.

2) Document Scheduling

Each time a document is sent to the Spooler, either from a process or on an input stream, parameters describing the document must be supplied. They are used to schedule the document, and are equivalent to the information found in the JOB card of other systems.

3) Document Control Commands

These commands are available to display the status of documents and allow their parameters to be changed once they are in the Spooling system. For example, a command is available to delete a queued document. There are different classes of access to documents for mainframe operators, remote operators and interactive users. It is also possible to enquire (for a limited period) about documents after they have left the Spooling system.

4) Operator Commands

Mainframe and remote operators require commands to control the streams in which their documents are being handled. Mainframe operators also require facilities to accept and refuse RJE log-ons.

6. Spooler Configuration

The Spooler configuration is an installation-dependent file of parameters which specifies to the Spooler which queues it has to maintain, the default parameters for documents put into these queues, and the streams which are to serve these queues. Streams are of various types: input and output communication streams to local peripherals, input and output streams to Remote Job Entry stations, streams to start batch jobs, and streams to send documents to other processes (e.g. JOBBER).

Queues are specified by giving them a name and a specification of default parameters for documents to be placed in that queue.

Examples

1) Defining a line printer queue:

```
QUEUE = LP                  ! The queue is to be known as LP.

USER = SPOOLR               ! Documents which do not have USER specified are
                                to be known as SPOOLR documents.

DELIVERY = SPOOLED OUTPUT   ! Documents with no delivery set are to
                                have this delivery.

START = 32                  ! Byte in file at which output is to start (by
                                default).

PRIORITY = 1                ! The default priority of the document.

TIME = NA                   ! This is not applicable to a line printer queue.

FORMS = 0                   ! Default forms required.

MODE = ISO                  ! Default output mode.

COPIES = 1                  ! Default number of copies to be output.

RERUN = YES                 ! Default is to reprint document if caught in a
                                System crash.
```

2) Defining a local line printer stream:

```
RMT = LOCAL          ! Location of device.

STREAM = LP0         ! Name of device.

STATUS = OPEN        ! Default status of device at SPOOLR startup.

NAME = X'06000000'   ! Adaptor no.: (dev type)<<24 ! (dev no.)<<16
                       etc.

QUEUE = LP           ! Queue to be served.

LIMIT = 256          ! Maximum size of document to be accepted, in
                       kilobytes.

FORMS = 0            ! Default forms type in printer.

MODE = ISO           ! Default output mode.

LOWEST PRIORITY = 1  ! Lowest document priority to be accepted.

TYPE = DEVICE OUTPUT ! Type of stream.
```

Note: Streams only accept documents when

```
        STREAM LIMIT     ≥ DOCUMENT LIMIT (SIZE or TIME)
  and LOWEST PRIORITY    ≤ DOCUMENT PRIORITY
  and STREAM FORMS       = DOCUMENT FORMS
  and STREAM MODE        = DOCUMENT MODE
```

Similar queues can be defined for batch jobs and documents queued for handling by other processes.

There are three types of output stream which can be defined:

1) DEVICE (document sent to a device)

2) JOB (document input to a batch process started)

3) PROCESS (document input to a running process, i.e. JOBBER)

Other parameters specified in the Spooler configuration include the number of documents which can be simultaneously queued and the number of document descriptors to be held per file system.

7. Document Scheduling Parameters

These are set by the user when he sends his document to be put into a Spooler queue or read into a file. Not all of these parameters are relevant for all queues, and defaults are supplied by the installation for each parameter where relevant.

Name	Use
USER	Name of the user owning the document.
PASS	Background password of user.
DEST	Name of queue or file the document is destined for.
SRCE	Name of file or device the document came from.
TIME	Cpu limit in seconds if document is a job.
DELIV	Delivery information of output from job or of file being output.
PRIO	Priority at which document is to be queued.
COPIES	Number of copies of document to be output.
FORMS	Form type required for output.
MODE	Mode of output: ISO, EBCDIC or BINARY.
ORDER	Order of job execution. Job run after all jobs for this user with value of ORDER < current run.
START	Byte in file at which output is to start.
LENGTH	Number of bytes to be output.
RERUN	Specifies whether document to be re-queued if caught in a System crash.
TAPES	Number of tapes required by the job.
DISCS	Number of discs required by the job.

Queue names are distinguished from filenames by their first character being a full stop.

Examples of document scheduling parameters

1) Sending a file to the line printer queue .LP:

```
DEST=.LP, USER=ERCC20, SRCE=LISTFILE, START=32, LENGTH=7194, DELIV=ERCC
```

2) Detaching a job to the batch job queue:

```
DEST=.BATCH, USER=ERCC20, SRCE=JOBFILE, TIME=75
```

When Documents are input from local or remote peripherals, the document scheduling parameters must be distinguished from the document. This is done by starting each line containing document scheduling parameters with //DOC and terminating the document with //DOCEND.

Example

Inputting a file to be listed on the line printer.

```
//DOC DEST=.LP, USER=ERCC20, DELIV=FRONT DOOR
    <file to be printed on the line printer>
//DOCEND
```

8. Document Control Commands

Once documents are queued by the Spooler for execution or for output, it is necessary for local (i.e. mainframe) operators, RJE operators and interactive users to be able to control the documents in the queues and enquire about their status. Below are described a set of commands to facilitate this. There are three classes of access to these documents:

1) Mainframe operators: complete access.

2) RJE operators: read access to all queues, but can only change documents queued for their RJE or sent from their RJE.

3) Users: read access and change access to their own documents only.

Commands	Params
COUNT	<rje><queue><docs>
FIND	<rje><queue><docs>
DISPLAY	<rje><queue><docs>
RUSH	<rje><queue><docs>
HOLD	<rje><queue><docs>
RELEASE	<rje><queue><docs>
DELETE	<rje><queue><docs>
MOVE	<rje><queue><docs> TO <rje><queue>
PRIORITY	<rje><queue><docs><numeric>
COPY	<rje><queue><docs> TO <rje><queue>
ROUTE	<rje><queue><docs> TO <rje><queue>

If command comes from

1) Mainframe operator: nothing is done

2) RJE operator: <rje> is set

3) A user: <username> is set

where:

<rje> = <rmtname> / <>

<queue> = <queuename> / <>

<docs> = <ident> / 'ALL' / <username>'.'<ident> / <>

Commands	Use
COUNT	Count all documents as specified by params. E.g.: 1) COUNT RMT25 counts all documents queued for RMT25 2) COUNT RMT25 BATCH counts all documents owned by RMT25 in batch queue 3) COUNT RMT25 BATCH ERCC20 counts all documents sent by RMT25 into the BATCH queue for user ERCC20.
FIND	Give summary of specified document.
DISPLAY	Display all information about documents as specified by parameters.
RUSH	Rush the document to the head of its queue.
HOLD	Hold the document specified in the queue.
RELEASE	Release the document specified from the held state.
DELETE	Delete the documents as specified.
MOVE	Move the documents specified to another queue.
PRIORITY	Set the priority of the documents specified.
COPY	Put copy of the documents specified in another queue but also leave them in the current queue.
ROUTE	Set the destination of output from a batch job.

9. Operator Commands

Some commands are available only to the mainframe operator, e.g. the accepting or rejecting of RJE log-ons, but other stream control commands are available to remote and local operators. Remote operators can only control streams connected to their RJE.

Mainframe only commands

Command	Parameters	Use
ACCEPT	<rje> or ALL	Allow RJE(s) to log on.
REFUSE	<rje> or ALL	Prevent RJE(s) from logging on.
WINDUP	<rje> or ALL	Allow current transaction to complete and log off RJE.
ABORT	<rje> or ALL	Log off RJE without finishing current transaction.
BROADCAST	<message>	Send message to all currently logged on RJEs.
NOTICEBOARD	<message>	Set message to be sent to all RJEs at log on.

Device commands for all operators

<device> = <RJE>'.'<devicename>/<devicename>

<device> FORMS = n <device> LIMIT = n <device> PRIORITY = n	Set device/stream characteristics.
<device> TO <queue>	Start device serving specified queue.
<device> ?	Find status of device.
BACKSPACE <device> FORWARDSPACE <device> INTERRUPT <device> RESTART <device> CONTINUE <device>	Device control commands not relevant to all devices.

W. Laing

Finding out about Spooler's Configuration from a Subsystem

To find out which queues are available to a Subsystem a file belonging to Spooler called SPOOLR.CFILE must be connected 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.

Relevant record formats and sizes

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.

QUEUES — specifies the size of the record array describing the queues in the configuration.

STREAMS — specifies the size of the record array describing the streams in the configuration.

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 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, i.e. bytes or seconds, currently queued. If DEFAULT TIME <= 0 then the amount is in bytes, otherwise the amount is in seconds.

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.

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, 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.

W. Laing

EMAS 2900 Spooler Note 3

The attached note describes the currently available facilities for file and job input to EMAS 2900. As the service develops, new facilities will become available, in particular new document scheduling parameters.

At the present time input to the 2970 is available from

1. EMAS 4-75 (The 2970 is known as .P49).

2. The card reader on the 2970.

3. The card reader and paper tape reader on the slow devices machine.

Input from JCMB Job Reception on level 3 and from Buccleuch Place Lane is currently under test. You are urged to try these new facilities and send any comments or suggestions to Bill Laing, Room 2020, JCMB.

EMAS 2900 SPOOLER NOTE

No: 3
Date: 8/5/79

Input of Files and Jobs to EMAS 2900

The Input of files and Jobs to EMAS 2900 is controlled by the paged System process SPOOLR. SPOOLR works with documents rather than with files. A document is made up of scheduling parameters, equivalent to a JOB statement on some other systems, plus the file to which they relate. SPOOLR is not interested in the contents of the file, only in the values of the scheduling parameters.

Using SPOOLR, it is possible to read files into disc storage and add files to System queues from cards or paper tape. This is usually achieved by submitting the file, with appropriate document scheduling parameters, to be read by a card reader or paper tape reader. In this case, an EMAS 2900 File Input Card should accompany the cards or paper tape(s). Alternatively, input can be sent from terminals connected to the medium speed network, and files can be transferred from other processors connected to the network. The details of these operations are subject to changes as the network is extended and modified; consult the HELP information for current facilities.

Card reader input

Cards can only be read in a mode involving translation from IBM 029 card code to ISO internal code. The file produced is an Edinburgh Subsystem character file. The following rules apply:

* All 80 columns are read.
* Trailing spaces are deleted; that is, all the spaces following the last non-space character on each card are excluded from the file.
* A Newline character is inserted in the file to signify the end of each card.

Paper tape input

Paper tape can only be read in one mode: ISO-coded 8-hole even-parity. The following rules apply:

* All characters with odd parity are converted to the SUB (ISO 26) character.
* The parity bit is set to 0 on input, whatever its value on the paper tape.
* Null (no holes punched) and Delete (all holes punched) are ignored.
* Carriage return characters are ignored when they are adjacent to newline characters.

* Trailing spaces (space characters immediately before a newline character) are ignored.

* All paper tapes must have at least 12 inches of run-out at both ends.

Remote processor input

Files are normally constructed on the remote processor with the document scheduling parameters included; they are then sent via the network to the EMAS 2900 System. The following rules apply:

* The final //DOC statement (see below) may be omitted from the file sent.

* The maximum size of file which can be read is 8 Megabytes; however, it may be lower for individual users (usually 1 Megabyte at present).

Scheduling parameters for file and job input

A document is input in the following form:

```
//DOC parameter=value,parameter=value,...

    contents of file

//DOC
```

The first DOC statement marks the start of a document and contains the scheduling information, specified as a series of parameter assignments; the relevant parameters are given in Table 1 below. Each assignment is terminated by a comma or newline (or end of card). If a line finishes with a comma, this implies that the DOC statement is continued on the next line. As many lines as desired may be used.

At least one space must follow the word DOC. Thereafter spaces are ignored unless they lie between the first non-space character and the last non-space character of the parameter value. Thus

```
//DOC PASS = 1234, U S E R = ERCC99,
    DEST= FILE, NAME= NEW1,
    DELIV = BLOGGS Chemistry
```

is equivalent to

```
//DOC PASS=1234,USER=ERCC99,DEST=FILE,NAME=NEW1,DELIV=BLOGGS Chemistry
```

Note that the DELIV parameter value can include spaces (as above), but not commas.

Parameter	Meaning
USER	The six-character username of the process receiving the file or executing the job.
PASS	The background password of the USER specified.
NAME	If the DEST specified is FILE it is the name to be given to the disc file to be created. It must be 11 characters or less, starting with a letter. A member of a partitioned file (an Edinburgh Subsystem file type) may not be specified. If the DEST specified is not FILE it is the name to be associated with the file as it is queued, executed or printed by the System.
DEST	The destination of the document. Table 2 gives the valid DESTinations which can be specified.
DELIV	Up to 31 characters of delivery information to be used on any output produced. If DELIV is not specified the default delivery for the user is used. Note that specifing DELIV does not change the user's default delivery.
TIME	The OCP time in seconds required by the job.
RERUN	Whether the job is to be rerun if caught in a machine crash. Should be specified as YES or NO.
COPIES	The number of copies of output required. Valid values 1-255.
FORMS	The forms type required for output. Valid values 1-255.
ORDER	The relative priority of execution of the job with respect to other jobs with the same DEST belonging to the same user. Thus, for example, if there are two BATCH jobs belonging to the same user in the system at the same time with ORDER values of 2 and 5, the job whose ORDER is 2 will be run first. Jobs must be submitted in the required running order. Valid values 1-255.
AFTER	The document is to be held in the queue until after the date and/or time specified. Valid values are of the form: date time time date date time where "date" is in the form dd/mm/yy, and "time" is in the form hh.mm.ss. If both "date" and "time" are given they must be separated by a space. See example (8) below.
FSYS	The file system on which the USER specified resides. This is not normally necessary: it is only required when the USER is accredited on more than one file system. Valid values 0-99.

Table 1: File and Job Input Scheduling Parameters

Notes

* Not all of the parameters given above have to be specified - there are defaults. The defaults can different for each document destination. If a file is to be created or a batch job executed, USER and PASS must be specified.

* Only one file can be created per //DOC ... //DOC set of statements.

* The //DOC statements must both start on a new line; this means that for paper tape input there have to be a line feed character before and after both //DOC statements.

* The NAME chosen for an input file should not be the same as the name of an existing disc file belonging to the specified USER. If it is, the existing file will be retained and the one being read in will be ignored.

* Files are CHERISHed automatically on input.

* Files may be placed immediately in output queues for printing or routing to a remote processor.

All valid output queues, i.e. LP, GP, LP15 etc.
JOBBER
BATCH
FILE

Table 2: Valid Document DESTinations

Examples

1)
```
//DOC USER=AAAA99,PASS = ABAB,NAME=DATAFILE23,DEST=FILE
  1  2  3  4  5  6
 72 341 896 247 839
 .
 .
 .
-1 -1
//DOC
```

2)
```
//DOC USER=ABCD01,DEST=FILE ,NAME=PROGA,PASS=XZ09
%BEGIN
    %INTEGER X,Y,Z
      .
      .
      .
%ENDOFPROGRAM
//DOC
```

```
3) //DOC DEST= BATCH,USER=AAAA99,PASS=BACK, NAME=BATCHJOB,
   TIME=20,DELIV=JCMB Room 2020
   DEFINE(1,.LP)
   RUN(PROGY)
   STOP
   //DOC

4) //DOC DEST=JOBBER
       .
       .
       JOBBER commands, etc.
       .
       .
   //DOC

5) //DOC DEST=BATCH,USER=AAAA99,PASS=BACK,ORDER=1,
       TIME=300
       .
       .
       .
   //DOC

6) //DOC DEST=BATCH,USER=AAAA99,PASS=BACK,ORDER=2,
       TIME=200
       .
       .
       .
   //DOC

7) //DOC DEST=BATCH, USER=AAAA99, PASS=BACK, ORDER=3,
       TIME=100
       .
       .
       .
   //DOC

8) //DOC DEST=BATCH,USER=AAAA99,PASS=BACK,NAME=AFTERJOB,
       AFTER=10/05/79 12.30.00
       .
       .
       .
   //DOC
```

Notes on the examples

1) Creation of disc file AAAA99.DATAFILE23; a file of the same name must not already exist. User AAAA99's background password must be ABAB.

2) Creation of disc file ABCD01.PROGA. Note that the space following FILE is ignored.

3) A batch job. Note that the delivery information (parameter DELIV) is not enclosed in quotes and that, because of the rules about spaces in DOC statements, the space characters in the delivery information are preserved.

4) Note all that is required (DEST=JOBBER) for JOBBER input.

5),6),7) If these three background jobs were submitted in that order, they would be run in the order specified via parameter ORDER - i.e. (5) then (6) then (7). If ORDER had not been specified, the order of running would have been (7) then (6) then (5), because of the TIME parameter settings; other things being equal, the shortest job will be run first.

8) This shows how a job can be held until after a specific date and time. Note that the use of AFTER does not guarantee that it will be run at the time specified, only that it will be run after this time.

W. Laing
J. Murison

EMAS 2900 SPOOLER NOTE

No: 4
Date: 20/10/79

Spooler Information Facilities for Subsystems

This is a brief description of how a subsystem writer can provide

(a) commands to find out about documents queued by SPOOLR, and

(b) commands to delete queued documents

There are currently three commands which can be issued by a subsystem:

QUEUE queuename, filename

DELETE document identifier

FIND document identifier, filename

where underlined parameters are obligatory.

These commands should be sent to SPOOLR using the DMESSAGE procedure. In the message string they must be preceded by "COMMAND ". For example:

```
MESSAGE = "COMMAND QUEUE BATCH,".BFILE
LEN = LENGTH(MESSAGE)
ERR = DMESSAGE("SPOOLR",LEN,1,-1,ADDR(MESSAGE)+1)
```

(See the EMAS 2900 Subsystem Writer's Manual, Appendix 1, for details of DMESSAGE.)

Other examples of messages which could be sent to SPOOLR via DMESSAGE:

COMMAND QUEUE ,FRED

COMMAND DELETE 010236

COMMAND FIND 020937,FRED

The general error flags which may be returned by these calls are:

201 Bad Params

214 Command Not Known

The parameters given should be of the form:

queuename — The name of a SPOOLR queue, i.e. up to fifteen characters long, not preceded by a "."

filename — A valid EMAS 2900 file name

document identifier — A six character numerical identifier, of which the first two characters specify the caller's file system.

Some comments about these commands:

QUEUE

This command returns information described below about all of the files a user has in SPOOLR's queues or in a particular queue. The filename specified is created in the callers file index and should not already have existed. Its format is described below. If the queue specified does not exist the effect is as if the caller had no files in the queue. The error flags returned by this command are:

15 No Free File Descriptors

16 File Already Exists

17 Insufficient Free Cells

18 Bad Filename

236 SPOOLR File Error

DELETE

This command takes only one parameter: a document identifier. The document specified must be in a queue for this command to take effect. For example, a job in execution or a file currently being printed cannot be deleted. The result of calling this command is passed back in the flag from DMESSAGE. The error flags returned by the command are:

205 Document Not Queued

213 Invalid Descriptor

236 SPOOLR File Error

FIND

This command takes two parameters both of which are mandatory: a document identifier and a file name. The comments about the filename given under the command QUEUE also apply here. The document descriptor may refer to a document which is not currently in a queue; for example, the document may be being executed or may have been deleted. This command currently can trace the last 1000 documents on each file system. The error flags returned by the command are:

15 No Free File Descriptors

16 File Already Exists

17 Insufficient Free Cells

18 Bad Filename

213 Invalid Descriptor

236 SPOOLR File Error

File returned by SPOOLR

The file returned by SPOOLR has the standard Edinburgh Subsystem header shown below. Note that if FILE HEADER_START = FILE HEADER_END then the file is empty.

```
%RECORDFORMAT FHF(%INTEGER END, START, SIZE, TYPE, SPARE0, %C
                  DATETIME, SPARE1, SPARE2)
```

The file returned contains (FILE HEADER_END - FILE HEADER_START)//256 entries. Each entry is 256 bytes, in the following format:

```
%RECORDFORMAT INFOF(%INTEGER VSN, STATE, %C
   %STRING (7) IDENT, USER, %C
   %STRING (15) DEST, SRCE, OUTPUT, %C
   %STRING (31) NAME, DELIVERY, %STRING (63) MESSAGE, %C
   %INTEGER DATE AND TIME RECEIVED, DATE AND TIME STARTED, %C
   DATE AND TIME OUTPUT STARTED, DATE AND TIME DELETED, %C
   DATA START, DATA LENGTH, TIME, OUTPUT LIMIT, PHYSICAL SIZE, %C
   PRIORITY, START AFTER DATE AND TIME, AHEAD, %C
   %BYTEINTEGER FORMS, MODE, COPIES, ORDER, RERUN, TAPES, %C
   DISCS, FAILS)
```

where:

VSN — The format version. Currently 1.

STATE — The state of the document. The states are:

0 Deleted
1 Queued
2 Sending
3 Running
4 Receiving
5 Processing

IDENT — The document identifier: two characters specifying the file system and four characters specifying a unique value.

USER — The username of the caller.

DEST — The name of the queue which the document is or was in.

SRCE — The name of the file or remote which the document came from.

OUTPUT — Not currently used.

NAME — The name the user gave to the document.

DELIVERY — The delivery information to be sent with the document when it is output.

MESSAGE — Not currently used.

DATE AND TIME RECEIVED — The date and time at which the document was placed in the queue.

DATE AND TIME STARTED — The date and time at which the document started execution or printing.

DATE AND TIME OUTPUT STARTED Not currently used.

DATE AND TIME DELETED The date and time at which the document was deleted.

DATA START If TIME is zero, the offset from the start of the file of the data which is to be output.

DATA LENGTH If TIME is zero, the number of bytes to be output.

TIME The job cpulimit in seconds.

OUTPUTLIMIT Not currently used.

PHYSICAL SIZE Not currently used.

PRIORITY The current priority of the document. Note that negative values specify that the document is HELD at the modulus of the value given.

1 VLOW
2 LOW
3 STD
4 HIGH
5 VHIGH

START AFTER DATE AND TIME The date and time until which the document is HELD in the queue.

AHEAD The accumulated bytes or seconds queued in front of this document.

FORMS The forms type required by this document.

MODE The mode which the data in the document is in. The values are:

0 - ISO
1 - EBCDIC
2 - BINARY

COPIES The number of copies to be output. Note that 0 means 1.

ORDER The order in which the document is to be removed from the queue, relative to other documents. Only relevant if non-zero.

RERUN Whether the document is to be requeued or deleted if caught in a System crash. 0 - No; 1 - Yes.

TAPES Not currently used.

DISCS Not currently used.

FAILS The number of times the document has failed in execution. Only relevant if RERUN = Yes.

W. Laing

EMAS 2900

SPOOLER 11I

Preview of changes/additions to spooler

Definites

i. Multiple queues served by one stream - the queues will be serviced equally in round robin fashion by the stream (as opposed to the current method of empty first queue, then second etc).

ii. Will allow trailing spaces on final //DOC card for a job.

iii. TAPES=n (where n is number of simultaneous decks required) will have the effect of holding the job and be job marked as a tape job in the batch queue.

iv. S/PRINT will send a copy to JOURNAL.

v. Spooler log will be limited to 128K.

vi. Fast version of Remove Zeros (from Kent) to be incorporated.

vii. Refresh will be removed or handled in a different way.

Possibles

i. S/DISPLAY nnnnnn
also will accept S/DISPLAY <USER>.<JOBNAME> <QUEUE>

John Henshall
4th June 1980

Intended implementation date 1.7.80

Any outstanding urgent requirements?

Mainframe only Spooler Commands

Informative Commands

K.WORD	K.WORD/PARAM
FE	<n(0,99)>
LAST	
NEXT	

Actions Commands (* See also Remote and Mainframe Spooler Commands)

K.WORD	K.WORD/PARAM	K.WORD/PARAM	K.WORD/PARAM
LOGON	< RNAME >	FE<n(0,99)>	< NADD >
LOGOFF	<RA>		
WINDUP	<RA>		
OPEN	TO	< RA >	
CLOSE	TO	< RA >	
TIE	DOWN	< RA >	
SWITCH	{ < RNAME > / { .ALL	TO / FROM	FE<n(0,99)> / FE<n(0,99)>
POLL			
CONNECTFE	<n(0,99)>		
DISCONNECTFE	<n(0,99)>		
SETMSG	<TEXT>|&<TEXT>		
BROADCAST	<RA>		
EOF	< SNAME >		
START	SA >		
STOP	<>|<SA>		
ATTACH	< SNAME >	TO	< QNAME >
DETACH	< SNAME >	FROM	< QNAME >
ABORT *	.ALL		
BATCHSTREAMS *	<n(0,16) >		
SELECT	< IDENT >		
TIDY	FSYS	.ALL |<n(0,99)>	
PRINT	<>|<QNAME>		
MON	< n(0,9)>		
PROMPT	ON/OFF		
CONFIG	<>|< FILENAME >		

Remote and Mainframe Spooler commands

Informative commands

K.WORD	K.WORD/PARAM	K.WORD/PARAM	K.WORD/PARAM
STREAM	<SNAME>		
S	<SNAME>		
STREAMS	<>|<n(0,255)>		
SS	<>|<n(0,255)>		
QUEUE	<QNAME>	<>|<USER>|<n(0,255)>	<>|<n(0,255)>
Q	<QNAME>	<>|<USER>|<n(0,255)>	<>|<n(0,255)>
QUEUES	<>|<n(0,255)>		
QS	<>|<n(0,255)>		
DISPLAY	<DOC>		
BATCHSTREAMS			
REMOTES	<>|<n(0,255)		
RS	<>|<n(0,255)		
REMOTE	<RNAME>		
R	<RNAME>		

Action Commands

K.WORD	K.WORD/PARAM	K.WORD/PARAM	K.WORD/PARAM	
ABORT	<SNAME>			
PRIORITY	<IDENT>|<SNAME>	<PRI>		
LIMIT	<SNAME>	n(0,)		
FORMS	<SNAME>	n(0,255)		
RUSH	<IDENT>			
HOLD	<IDENT>			
RELEASE	<IDENT>			
DELETE	<IDENT>|<USER>.ALL|<QNAME>.ALL	<>|<QNAME>		
MOVE	{ <IDENT>|<QNAME>.ALL { <USER>.ALL	TO <QNAME>	<QNAME> TO	<QNAME>
COPY	<IDENT>	TO	<QNAME>	
OFFER	<QNAME>			
WITHDRAW	<QNAME>			

Note : When issued by a remote operator the 'Action Commands' given above will only be permitted where

i) the Stream<SNAME> is 'owned' by that remote.
or ii) the document<IDENT> either
a) originates from that remote
or b) resides in a queue that is served by a stream 'owned' by that remote.

In the case of MOVE and COPY the receiving queue must be either 'owned' by the actioning remote or the 'owning' remote must have 'OFFERed' the queue.

Note : local line printer is always 'OFFERed' in this sense.

Remote only Spooler Commands

Action Commands

K.WORD	K.WORD/PARAM	K.WORD/PARAM
LOGON	< RNAME >	<PASS >
LOGOFF		
WINDUP		
MSG	< TEXT >	

&	
ON	
OFF	
.ALL	LITERALS
FE	

<RINTEGER>	INTEGER IN A SPECIFIED RANGE
<FILE>	A valid file name
<USER>	A valid user name
<QNAME>	A valid queue name
<SNAME>	A valid stream name
<RNAME>	A valid remote name
<IDENT>	A valid document identifier
<TEXT>	A valid piece of text
<PASS>	A valid remote password
<NADD>	A valid network address
<>	No qualifier

<n(x,y)>	<RINTEGER>
<FILENAME>	<FILE>/<USER>.<FILE>
<SA>	.ALL/<SNAME>
<QA>	.ALL/<QNAME>
<RA>	.ALL/<RNAME>
<DOC>	<IDENT>/<USER>.<FILE>
<PRI>	VLOW/LOW/STD/HIGH/VHIGH

3.4 Spooler Commands

The EMAS Spooler (process name SPOOLR) is responsible for the administration of all "queues" within the System (e.g. batch input, printer output etc.). The queues are defined by a configuration file read when the SPOOLR process starts up; the name of the current configuration file is displayed on the OPER at that time.

Commands to SPOOLR are input in the form:

S/keyword parameter

The following is divided into the various areas of activity which Spooler controls; namely Stream Control, Queue Control, general commands, and Remote Station Control.

Stream Control

These commands manipulate "streams"; each "stream" serves one or more output queues, or one input queue. Where the word "stream" appears in a command, it may be replaced by the name of a device or stream (e.g. LP0, JOB0) or, except where noted otherwise, by ".ALL" in order to specify all streams.

ABORT stream - abort the current transaction on the specified stream without allowing it to complete; note that the document is requeued and restarted immediately, since the stream will not actually be stopped.

Example: S/ABORT LP23

ATTACH stream TO queue

- cause the specified stream to serve an additional queue. If a stream is attached to more than one queue, documents are taken from each queue, in rotation. Only one queue may be associated with an input stream.

  It is used, for example, to send documents queued for "LP" to the device "LP23" rather than the local "LP0" if "LP0" is down.

  Example: S/ATTACH LP23 TO LP

DETACH stream FROM queue

- stop the specified stream from serving the specified queue.

  Example: S/DETACH LP23 FROM LP

EOF stream - terminate binary input.

Example: S/EOF CR0

FORMS stream n - set the forms type for the stream to n. Note that documents are only sent down streams when the forms they require are the same as those set on the stream. Also note that 0 <= n <= 255. The ".ALL" form may not be used.

Example: S/FORMS GP23 1

LIMIT stream n - set the limit for single documents on the stream to n, which must be >=0. n may be an output limit in bytes, an input limit in seconds, etc. The ".ALL" form may not be used.

Example: S/LIMIT JOB1 600

PRIORITY stream/ident priority

- set the specified priority on the stream or doc. Note that, currently, the VLOW priority must be used at all times. The ".ALL" form may not be used. "priority" may be one of the following:

VLOW, LOW, STD, HIGH, VHIGH

Examples: S/PRIORITY LP0 VLOW
S/PRIORITY 000569 VHIGH

START stream - start the specified stream. The stream will then serve its queue as required. This command must be given after Spooler has started, as most streams are initially stopped; line printer paper may thus be lined up before printing starts. The LIMIT command (see below) can be used in conjunction with the START command.

Examples: S/START .ALL
S/START LP14 LIMIT 100

STOP stream - abort the specified stream (if active) and stop any further activity on that stream, (but for STOP with no parameter, see under "General Commands" below).

Example: S/STOP LP23

WINDUP stream - set the limit to zero on the specified stream(s) so that no new activity takes place on that stream, although the current activity is allowed to complete.

Example: S/WINDUP LP14

Queue Control

The commands manipulate items ("documents") within Spooler's queues. In the following commands, the mnemonic qudoc can be replaced by any of the following:

ident	- is the full six digit unique identifier for the queue item, e.g. 020043.
queue ident	- is the queue name followed by the unique identifier.
queue user name	- is the queue name, the user who owns the document, and the document name.

COPY ident TO queue- make a COPY of the document and add it to queue.

Example: S/COPY 000376 TO LP23

DELETE qudoc or
DELETE user.ALL queue or
DELETE queue.ALL

- delete the specified document from its queue. The document is effectively destroyed.

Example: S/DELETE 100562
S/DELETE ERC009.ALL LP23
S/DELETE LP23.ALL

DISPLAY qudoc - display full details of the specified document.

Example: S/DISPLAY 090724

HOLD qudoc - hold the specified document in the queue until it is DELETEd or RELEASEd.

Example: S/HOLD LP14 ERCC02 TESTFILE

MOVE qudoc TO queue or
MOVE queue.ALL TO queue or
MOVE user.ALL queue TO queue

- move the specified document to a different queue.

Examples: S/MOVE 100562 TO LP
S/MOVE LP23.ALL TO LP
S/MOVE ERC001.ALL LP23 TO LP

The receiving queue must be either 'owned' by the actioning remote or the 'owning' remote must have 'OFFERed' the queue. This restriction does not apply if the action comes from the mainframe.)

RELEASE qudoc - release the specified "held" document, so that it may leave the queue.

Example: S/RELEASE 200401

RUN <strm> <doc> - forces stream <strm> to run job <doc> regardless of BATCHSTREAMS setting and stream limit, but subject to all other checks; e.g. PRIORITY, AFTER.

Example: S/RUN JOB2 000469

RUSH qudoc - set the priority of the specified document to maximum.

Example: S/RUSH 210211

SELECT qudoc - select a "media" batch job to run. A "media" batch job will be held by Spooler, indicating to the operators the media required; e.g. 2MTU, 1EDU. The SELECT command gives the operator the ability to schedule media jobs.

General Commands

The commands specified below are not specific to stream or queue control. The mnemonics used are as follows:

onoff	- means ON or OFF.
pageno	- means a page number for display; may be omitted for the first page (page 0).
queue	- means a queue name.
stream	- means a stream name.

BATCH - display the current batch stream settings.

BATCHSTREAMS n lim

- set the maximum number of batch jobs which may be run at once. If n is set to 0 no further batch jobs will be started.

  Example: S/BATCHSTREAMS 3

  An overriding upper limit for any job started can be set (lim):

  Example: S/BATCHSTREAMS 7 30

  This would allow up to 7 streams to run jobs of no more than 30 seconds.

CONFIG n filename - set the name of the configuration file to be used by Spooler on Fsys n when it is next started. The Spooler prompt should be ON when using this command (see below).

  Example: S/CONFIG 0 MANAGR.NEWCONF

  If filename is omitted the current configuration filename will be printed.

FEPUP - re-establish communication with a Front End which has crashed and been re-IPLed.

FEPDOWN - clear down communications with a Front End.

LAST - page backwards the current Spooler display on OPER.

MON n - switch internal Spooler monitoring on. n=0-9 depending on the level of monitoring required.

NEXT - page forward the current Spooler display on the OPER.

PRINT - send copies of Spooler's log to the line printer queue and to the JOURNL queue.

PROMPT onoff - PROMPT ON causes Spooler to prompt for further input. The prompt is "SPOOLR:". The "S/" may then be omitted from further input, and the "COMMAND" key should not be pressed unless input is to go to a process other than SPOOLR. PROMPT OFF causes the "SPOOLR:" prompt to be discontinued.

QUEUE [user.ALL] queue [pageno]
or Q [user.ALL] queue [pageno]

- (items in [..] brackets are optional) display the specified page (or page 0) of the specified queue for the specified user (or all users if none specified). The long form (QUEUE) gives additional information about streams attached to the queue, defaults, etc. The short form (Q) does not list AFTER documents that have not yet become eligible to be released by Spooler; they will only be listed by QUEUE.

Example: S/QUEUE LP23
S/Q BATCH
S/Q LP 1
S/Q ERC009.ALL LP23

QUEUES pageno
or QS pageno - display Spooler's queue, which contain files, jobs, etc. Totals (KBYTES or SECS as appropriate) are also given. The long form (QUEUES) shows all queues; the short form (QS) shows only the non-empty queues.

Examples: S/QUEUES 1
S/QS

STOP - stop all the active streams, when they are all idle, the Spooler process is stopped.

STREAM stream
or S stream - display the current status of the specified stream. If the stream is currently active, or has just finished being active, details of the document being processed are displayed. There is currently no difference in the action taken for the two alternative forms of this command.

Examples: S/STREAM LP23
S/S LP23

STREAMS pageno
or SS pageno - display all streams, and which process they are active for. The long form (STREAMS) shows all streams; the short form (SS) shows only the active streams.

Examples: S/STREAMS
S/SS

Commands for Controlling Remote Stations

The commands specified below are used to control remote terminals for remote job entry, input-output, etc. The definition of "remote" is given first.

remote - means a remote terminal name. Where this appears, it may be replaced by ".ALL" in order to specify all remotes.

BROADCAST remote - send the text message stored by the MSG command to remote.

Example: S/BROADCAST T15

CLOSE remote - refuse logons from the specified remote(s).

Example: S/CLOSE .ALL

LOGOFF remote - force logoff of the specified remote(s).

LOGON remote FEn network address

- should be used to log on a pre-logged on terminal after the terminal has been logged off. A pre-logged on terminal can be moved to a different Front End by logging it off and then logging it on again.

Example: S/LOGON T80 FE2 80 80

MSG text - store a text message for subsequent BROADCAST to remotes.

OPEN remote - accept logons from the specified remote(s).

POLL - initiate the polling of the FEPs to decide whether any FEP is a better 'route' than the current 'logged on' FEP for any remote; if so the remote is cleanly SWITCHED to this FEP. Polling is done automatically approximately every half hour.

SWITCH remote TO FEn or
SWITCH remote FROM FEn

- force a remote to switch from its current FEP to the defined FEP.

Example: S/SWITCH T14 TO FE2

TIE remote - tie down a remote to its current FEP.

Example: S/TIE T14

REMOTE remote
or R remote - display the current status of the specified remote There is currently no difference in the action taken for the two alternative forms of this command.

Examples: S/REMOTE T14
S/R T14

REMOTES pageno
or RS pageno - display the status of all remotes (for the REMOTES command) or all logged-on remotes (for the RS command).

Examples: S/REMOTES 1
S/RS

WINDUP remote - set the limit on all streams on the specified remote to zero, so that no new activity may take place on it. Current activities are allowed to complete. Further activity may be allowed to complete. Further activity may be allowed by means of the LIMIT command.

Example: S/WINDUP .ALL

Remote-only Spooler Commands

There are a number of commands which are only relevant to remote terminal operation.

DISABLE - disable the operator console from inputting any further commands.

ENABLE - enable the operator console for input after the DISABLE command. A password is prompted for at this stage.

OFFER queue - offer the queue to all other remotes for transferring documents to this queue. In the case of the MOVE and COPY commands the receiving queue must be either 'owned' by the actioning remote or the 'owning' remote must have OFFERed the queue.

WITHDRAW queue - withdraw the OFFER.

Subsystem

EMAS 2900 SUBSYSTEM WORKING NOTE

No. 1
Date 5/4/77

1.1 Introduction

This is the first in a series of notes which will describe features of the initial subsystem being implemented on EMAS 2900 There is a need to satisfy two conflictin requirments.

(1) To provide a stable base for Subsystem users - including VOLUMES and JOBBER

(2) To retain freedom to alter the specification in detail.

As a compromise I propose to provide with each routine specification an indication of its likely stability. In the following notes it can be assumed that EMAS rules apply unless otherwise indicated.

1.2 File names

Initially I propose that we stick to EMAS1 file name limit. These are:-

User (6) . file (8) _ member (8)

This requires 24 characters. For various reasons it may be desirable to extend these fields so a compromise length of 31 is suggested for all filename parameters. In fact the 2900 IMP parameter passing mechanism does not put strings on the stack it puts a descriptor there, so precision is not as important as on current EMAS.

1.3 Subsystem structure

There will be a small kernel of routines and tables to organise the virtual memory and provide minimal character output - probably only one stream. This will be used by the SPOOLER and JOBBER and later by the rest of the Subsystem. My initial aim is to get this kernel working reliably. The routines provided in it will be very similar to their counterparts on System 4.

1.4 Basic File manipulation

The processes of creating, destroying, connecting and disconnecting are much as on EMAS 1. Director does not yet support changes to connect mode,nor changing size of connected files. These activities will be provided by the subsystem using disconnect and connect. Where necessary the re-connection will be back at the same address. The first version of the kernel will contain the following file manipulation routins:

CONNECT
DESTROY
DISCONNECT
FILELIST
FINFO
OUTFILE

1.4.1 CONNECT

Specification:

%SYSTEMROUTINE CONNECT (%STRING(31)FILE, %INTEGER MODE, HOLE,
PROTECT, %RECORDNAME R, %INTEGERNEME FLAG)

MODE= 1 for write 0 for read

HOLE= connect hole - in bytes - default is current size of file

PROTECT= 0 normally

R has following format

%RECORDFORMAT RF (%INTEGER CONAD, FILETYPE, DATASTART, DATAEND)

FLAG values will be supplied later.

Note that this routine can also be used to connect a member of a partitioned file.

Stability - changes of detail likely.

1.4.2 DESTROY

Specification:

%SYSTEMROUTINE DESTROY (%STRING(31)FILE, %INTEGERNAME FLAG)

File will be disconnected (if necessary) and destroyed. Can also be used to destroy a member of a partitioned file.

Stability - unlikely to change.

1.4.3. DISCONNECT

Specification:

%SYSTEMROUTINE DISCONNECT (%STRING(31)FILE, %INTEGERNAME FLAG)

Use obvious. Not applicable to member of partitioned file.

Stability - unlikely to change.

1.4.4 FILELIST

Specification:

%SYSTEMROUTINE FILELIST (%INTEGER MODE, AD1, AD2)

Get list of files in format according to mode. AD1 and AD2 used to provide addresses of areas in users program for information. Initially:

MODE= 1

AD1 points to start of %STRING (8) %ARRAY

AD2 points to an integer into which is put the number of files.

Stability - significant changes and enhancement intended.

1.4.5 FINFO

Specification:

```
%SYSTEMROUTINE  FINFO (%STRING(31) FILE, %INTEGER MODE,
    %RECORDNAME  R,  %INTEGERNAME FLAG)
```

R has following format:

```
%RECORDFORMAT RF (%INTEGER CONAD, FILETYPE, DATASTART, DATAEND, SIZE, RUP, EEP,
MODE, USERS, ARCH, %STRING(6) TRAN, %STRING(8) DATE, TIME  %INTEGER COUNT, SPARE1,
SPARE2)
```

For MODE= 0 the following fields will be filled:

SIZE, RUP, EEP, MODE, USERS, ARCH, TRAN

Stability - significant changes and enhancements intended.

1.4.6 OUTFILE

Specification:

```
%SYSTEMROUTINE  OUTFILE (%STRING(31)S, %INTEGER SIZE, HOLE, PROTECTION,
%INTEGERNAME CONAD, FLAG)
```

File will be created with header containing default values: 32, 32, sized, 0,0,0,0,0
(see 1.5 - below) The rest of the file will be cleared

Stability - stable apart from minor details.

1.5 File Format

1.5.1 The file header format will be as follows:

Length= 32 bytes = 8 words used as follows:

Word	
0	Total length of header and data
1	Length of header (default 32)
2	Physical size
3	Type = 0 non-standard 1 object 3 character 4 data file (discrete records) 5 partitioned
4	Data last altered (packed)
5	Time last altered (packed)
6	Type dependent
7	Type dependent

1.6 Ancilliary routines

1.6.1 MOVE

Specification:

```
%SYSTEMROUTINE  MOVE (%INTEGER BYTES, FROM, TO)
```

Move 'BYTES' bytes from address 'FROM' to address 'TO'

Stability - stable

1.6.2 FILL

Specification:

%SYSTEMROUTINE FILL (%INTEGER BYTES, START, PATTERN)

The least significant byte in 'PATTERN' is copied into the 'BYTES' bytes starting at 'START'

Stability: stable

1.6.3 DUMP

Specification:

%SYSTEMROUTINE DUMP (%INTEGER FROM, TO)

The area 'FROM' to 'TO' is dumped on the current output stream.

Stability stable

1.6.4 PHEX

Specification

%SYSTEMROUTINE PHEX (%INTEGER N)

The value of N is printed as 8 hex. digits

Stability stable

1.6.5 PON, POFF, OUT

These all take one parameter of type %RECORDNAME with format:

%INTEGER DEST, SRCE, A, B, C, D, E, F

1.7 Character output

The initial version of Kernel will allow for character output to a single steam - the line printer. Output will be put in a file which can be listed an any time by calling systemroutine CLOSEST. The effect of this routine will be to list the current output file and reset the output pointer to the start of the file for further output. (c.f. System 4 VOLUMES)

Roderick McLeod

Existing and proposed developments

2.1 Introduction

This note, which follows 3 months of development of the Subsystem, contains details of additional facilities and some discussion of proposed developments. One major change has occurred since the first note: VOLUMS is no longer based on the standard Subsystem. Whilst this potentially increases the amount of software to be maintained it significantly simplifies the Subsystem in that it no longer has to be completely general in the file handling area.

Currently Alan Anderson is working on mounting the Scientific Jobber on EMAS and this will use, initially at least, a kernel of routines from the standard Subsystem. The future of this development depends on some difficult decisions about compatibility between foreground and batch access.

2.2 Filenames

A "final" decision has been reached concerning the length of filenames. The new form is

user(6).file(11)_member(11)

Although I have reservations about this decision I accept that it has its good points - such as the fact that the maximum length of a filename will fit into the string (31) parameters I proposed for filenames in my last note. All existing Subsystem software has been adjusted to allow for this change.

2.3 Foreground commands

Currently the following foreground commands are implemented:

ACCEPT	
ALGOL	
COPY	(as COPYFILE)
CPULIMIT	
DEFINE	(first parameter is channel no. - e.g. DEFINE 1,,LP)
DESTROY	
DISCONNECT	
EDIT	
FILES	(as FLIST)
FORTRAN	(as FORTE)
IMP	
LINK	
LIST	
LOOK	
METER	
OBEY	(as OBEYFILE)
NEWGEN	

OFFER
PARM
PERMIT (as PERMITFILE)
RENAME
RUN
STOP

In general commands operate as on System 4 EMAS. There are a few restrictions and differences - details are available from me. Many of the hash commands are also available - details later.

2.4 Data handling

2.4.1 Character Files

Character I/O for IMP and ALGOL has been implemented using a completely rewritten IOCP. Its main differences are as follows:

* no line reconstruction - hence no trailing space deletion nor detection of SUB character.

* no margins - the compilers currently do their own input framing

* stream selection no longer works on a record basis - it can now be done between individual characters

2.4.2 Data Files

Currently I am working on this. On its own it is reasonably straightforward. The Fortran I/O package from Geoff's group has fairly well-defined interfaces. The snags concern mixed access to the same file or device - e.g. the input teletype being used both to read commands (the command interpeter being written in IMP of course) and for input to a Fortran program. The switch between the two is not easy to organise in an elegant way.

2.5 Compatibility.

I am constantly considering the implications of compatibility with System 4 EMAS. Although there is the possibility of external use of EMAS 2900, my primary interest is still in its use by our current EMAS fraternity. I am quite sure that the quest for compatibility must not be allowed to hamper development at this time but there are obvious advantages in allowing a straightforward transfer of work from the System 4 to the 2900. There are two parts to the compatibility problem:

* Changes to foreground commands affect everyone but are probably fairly easy to get used to.

* Changes to commands called from programs mean alterations to existing working programs. This will involve significant work for people transferring such programs from System 4 to 2900.

2.5.1 Compatibility of files

Currently the position is as follows:

Type	Compatibility
Character	Vital - no great problem.
Data	V format files will not be compatible - a conversion utility should be straightforward. F format files OK.
Object Files	Not needed.
N.R. Object Files	Currently compatible but ought to change to conform to standard file header format.
Library Index files	Not compatible.
Partitioned files	Almost compatible - existing directory format has room for 11 character member names. Header format not quite right.

2.5.2 Compatibility - other features

One difficult one is .TT. Most of us would find great difficulty in defining our interactive terminals other than as .TT I suspect. Nevertheless it is an outdated abbreviation. One possibility is something like .IT for 'interactive terminal' - but currently I propose to introduce the pseudo devices .IN and .OUT. These would direct input and output to the current main input and output devices. During foreground sessions this would normally be the interactive terminal - unless an OBEY command was being executed.

2.6 Conclusion

As always I welcome comments on the above ideas.

Roderick McLeod

EMAS 2900 SUBSYSTEM NOTE

No: 3 (revised)
Date: 11/5/79

User process information

The SSINFO facility provided on the System 4 has proved to be a clumsy solution to the problem of providing information about a user's process, in that the user has to provide a long record format even when he only requires one minor item. The equivalent information on the 2900 is provided by a pair of functions: a string function for textual information, and an integer function for integer information.

They have to be specified:

```
externalstringfnspec    UINFS (integer ENTRY)

externalintegerfnspec   UINFI (integer ENTRY)
```

The parameter ENTRY in each case specifies which piece of information is required. If a value is given which is out of the current range then the result will be a null string or zero, as appropriate. Currently the following values of ENTRY can be used:

UINFS

1	User name
2	Delivery information
3	Session start time hh.mm.ss
4	PROMPT text
5	Active directory
6	Subsystem version

UINFI

1	File system number
2	Mode: result = 1 implies foreground result = 2 implies background result = 3 implies foreground OBEY
3	Users currently logged on
4	ACR level
5	Current CPU limit (seconds)
6	Max size in Kilobytes for DEFINE
7	SYNC1DEST
8	SYNC2DEST
9	ASYNCDEST
10	Address of Director USERINF record
11	Process number
12	DEFINE level
13	Process incarnation number
14	AUXSTACK size (Kilobytes)

Further entries will be added as required.

R.R. McLeod

New Facilities

Date and Time

I have had a few comments about my proposal (System Note 5) for packed date and time. Apart from treating bit 2**31 as the most significant bit of 'year' (hence giving a range 1970-2033), I propose using the format in the Subsystem and will be changing to it during the next few weeks. FILEANAL will only give the time and date for recently altered files.

System Routine ANALYSE

To enable users to more readily write their own file analysis routines I am providing a new general-purpose systemroutine ANALYSE. This will return information in a recordarray for the user to analyse. It will be useful for Object, Directory and Partitioned files in particular. The standard command FILEANAL will call this new routine. Full details later.

New Command ALIAS

This new command allows you to add an alias to a procedure name. The format is

```
ALIAS(procname,alias)
```

Hence if you want to access the command (procedure) IMP using the alias I, you should type

```
ALIAS(IMP,I)
```

To remove all the aliases of a given name omit the second parameter; e.g.

```
ALIAS(IMP)
```

String Function INTERRUPT

A new externalstringfn INTERRUPT is available. It replaces TESTINT. It has no parameter, and returns and clears the latest interrupt of more than one character from the user terminal. A null string is returned if no multi-character interrupt has occurred or if the last one has already been returned by INTERRUPT. Note that multi-character interrupts are not queued. Single-character interrupts are reserved for Subsystem use.

Example:

```
INT = INTERRUPT
if INT = "STOP" then return
if INT = "RESET" then .........
```

Directory Files and Related Commands

Directory files serve a similar role to library index files on the 4-75. Their functions are:

* For each procedure entry name or data entry name in a directory, to provide a pointer, either to the name of the object file that contains the entry, or to another directory (from which the name of the appropriate object file can be found, directly or indirectly).

* To hold the ALIAS information (see above).

NEWDIRECTORY

This command is used to create a new directory file. Only one parameter is required, the name of the directory. Currently this must be 'SSWDIR' since the other commands only operate with respect to this directory; later it will be possible to create other directories. Two further parameters can be used to specify the size of the directory - see Mac for details.

INSERT

This command is similar to INSERTFILE on the 4-75. Note however that it has to be called explicitly once for each object file. Once it has been called for an object file, that object file will be REMOVEd from and INSERTed into the currently selected directory every time it is compiled into. If a compilation into the file fails, only the object file name will remain "inserted", but all the information will be put back when it is successfully recompiled.

INSERT has only one parameter - the name of the object file.

It is not necessary to INSERT main program object files.

REMOVE

This command is the equivalent of REMOVEFILE on the 4-75. Only one object file can be removed at a time.

Program Loading

The program loader can now handle external data linking. Note that a check is enforced on the lengths of external and extrinsic items. The order of satisfying references differs from that on the 4-75. The change has been made in the interests of efficiency, and does currently restrict the generality compared with the 4-75. If necessary I shall provide additional facilities to enable users to override standard command names. The mechanism works as follows:

* There is a session directory of all currently loaded entries; at process start-up this includes only the standard commands which are in the basefile.

* If a standard command is accessed which is outwith the basefile, e.g. one of the Subsystem editors or a compiler, then the necessary files are loaded for the rest of the session and their entry information is added to the session directory.

* If a user-written command is accessed, it is loaded, and then unloaded on return to command level. A facility will be provided to nominate user object files which are to be loaded for the whole session.

* When searching for a procedure name the loader first searches the session directory, then the user's own SSHDIR (if it exists), then the Subsystem base directory. There are two implications:

  * it is not possible to override commands in the basefile

  * it is not possible to override other Subsystem commands if they have been accessed earlier in the session

For example, if you wish to use a new IMP compiler you must insert it in your directory. Even so you will not be able to access it if you have already used the standard IMP compiler during the session. If you log off and then on again you will obtain access to the new compiler.

Current Developments

1. Extensions and improvements to FILEANAL.

2. Provision of an OPTION command. This will incorporate APPENDLIB.

3. Provision of full I/O facilities for IMP and FORTRAN.

Roderick McLeod

Edinburgh Subsystem Standard File Formats

Introduction

This note supersedes all earlier descriptions of file headers. It contains information about the contents of headers of files produced by the Edinburgh Subsystem. It also contains some comments on backward compatibility with 4-75 files. Object files still have a format which does not fit in with other types. I am awaiting changes to LPUT from Geoff Millard, but I have to warn that there will be considerable upheaval when the change is made.

Information common to all files

The first six integers are used as follows:

1. Useful length of file: this is the sum of the length of the header and the length of the data (bytes).

2. Length of header (bytes).

3. Physical size of file: this will normally be the total size of the minimum number of full pages needed to hold the useful length (bytes).

4. File type: 1 - Object file (eventually)
   2 - Directory file
   3 - Character file
   4 - Data file (discrete records)
   5 - Partitioned file

   These are also the values returned by CONNECT.

5. This word is reserved for an optional sum-check. The facility of sum-checking vital files on the 4-75 (e.g. compilers) has proved to be very useful. A similar but standardised scheme is envisaged here; further details later. Note that in the case of fixed-up Director and Subsystem files this word may have to be used for a PC relative jump, since the Local Controller and Director respectively cannot readily read an offset from file; so always set the PC to 16 bytes from the start of the file. There is no obvious resolution to this.

6. Packed date and time of last writing to the file. This is updated whenever the file is connected by the Subsystem in write mode. The format is described in EMAS System Note 5.

Integers 7 and 8 are used in type-specific ways, as follows:

Type	7	8
Directory	Offset of P list (Note a).	Spare
Character	Spare	Spare
Data	Record format and maximum or fixed record size (Note b).	No. of records
Partitioned	Offset of directory of members (Note c).	No. of members

Notes

a) The internal format of a directory file will be published later. Briefly a directory comprises two parts: a hashed table of entry names and aliases, each of which points to a name in the "P list". Additionally, for long names in the hashed table the first part of the name goes in the table, together with a pointer to the rest of the name in the P list.

b) The format of this integer is

(R<<16)!(F)

where R is the maximum or fixed record length and F is 1 (fixed), 2 (variable) or 3 (unformatted). An unformatted file is the equivalent of an SM file on the System 4.

c) The directory of members is an array of records with the following format:

```
recordformat PDF (integer START, string (11) NAME, c
                  integer HOLE, S1, S2, S3)
```

Backward compatibility to 4-75

In the short term files will be transferred between the 4-75 and 2900 via controlled routes, and utility programs can be provided on each machine to carry out the necessary conversions. Eventually it is proposed to have a common format for archive and backup tapes between the systems. The desirable aim of maintaining identical file formats between the two Subsystems has been sacrificed in the interests of a tidier situation on the new Subsystem. In detail, only three types of file can sensibly be transferred:

Character	Minor change: dual standard could be accommodated in both Subsystems.
Data	Significant changes: will require conversion program, which will only be made available on the 2970.
Partitioned	Minor change: will require a conversion program; also a problem of individual members having the "wrong" format; also a backward transfer problem in that member names of up to 11 characters are allowed on the 2970, while only 8 characters are allowed on the 4-75.

Summary

Even with a common format for System archive and backup tapes the ready transfer of files between Systems, other than character files, will not be achieved.

Roderick McLeod

Diagnostic Aids in the Edinburgh Subsystem

The following commands are available for diagnostic purposes; in the main they replicate facilities on the 4-75. Note that:

* The fact that they are described here should not be taken as a guarantee of their long term support.

* In all relevant cases lengths are expressed in bytes.

Command structure

Each command has a # as its first character. Note that hash commands do not invoke the Loader and there is no one-to-one correspondence between each hash command and an external routine. Therefore hash commands cannot be called from programs.

The parameters to hash commands include numeric constants. These can be typed as decimal integers in the normal way, or as hexadecimal numbers, in which case they should be preceded by an 'X'. For example, the following two commands would have the same effect:

```
#SWORD(X840000,256)

#SWORD(X840000,X100)
```

Individual command descriptions

The hash commands available are as follows:

#ACR

prints out the current ACR level.

#CONNECT(filename)

connects the file, if possible in write mode, otherwise in read mode. Prints the connect address. Can also be used for a member of a partitioned data set, but the member is always connected in READ mode. Since a file connected by #CONNECT remains connected on return to command level, it is advisable to disconnect it explicitly when any operations upon it have been completed.

#DEC(hex value)

converts the hexadecimal value to decimal.

#DUMP(address, bytes)

dumps the specified area to the line printer.

#DUMPFILE(filename, offset of start, bytes)

dumps the specified area of the file to the line printer.

#HEX(decimal value)

converts the decimal value to hexadecimal.

#PCOM(integer)

prints out the value of the specified location in COMREG.

#PMESS(integer)

prints out the Subsystem error message associated with the specified fault number.

#PVM

prints table of connected files.

#QUIT

logs off even when the session directory is corrupted.

#REGS

prints out registers at the time of the most recent failure contingency. Note that information on the last four contingencies is held; the three previous ones can be obtained using #REGS(-1), #REGS(-2) and #REGS(-3).

#SBYTE(address,value)

sets the specified byte to value. Clearly the byte must be accessible in write mode. This can be achieved by connecting the file using #CONNECT.

#SCOM(integer,value)

sets the specified location in COMREG to value.

#SETBASE(filename)

sets the name of the basefile to be used for subsequent sessions. If the parameter is omitted the default Subsystem basefile will be used.

#SNAP(address,bytes)

dumps the specified area on .OUT. The amount printed on a line will depend on the current setting of OPTION ITWIDTH.

#SNAPCODE(address,bytes,out)

decompiles code and prints it on the device or file specified. Default is .OUT.

#SNAPCH(address,bytes)

prints the specified area as ISO characters on .OUT.

#SSTRING(address,"string")

sets the string at the address specified to the string given. Note that the string should be enclosed in double quote characters.

#SWORD(address,value)

sets the word at the address specified to value.

Additional commands will be added in the light of experience.

Roderick McLeod

DOC/EMAS.K2.0/1

University of Kent at Canterbury	COMPUTING LABORATORY
Diagnostic Aids in the Edinburgh Subsystem R D Eager	
Date August 1979	Latest Revision March 1984 (2)

Contents

1 Introduction

2 Command Structure

3 Individual command descriptions

4 Error messages

5 Warning

6 Acknowledgement

University of Kent at Canterbury © 1979

1 Introduction

The commands described in this document are provided for low-level diagnostic purposes; as such, they are of interest mainly to system programmers and anyone else using low-level facilities in the Edinburgh Subsystem.

A few commands are of wider interest; for instance, those concerned with loader monitoring and timing.

In general, if a user cannot understand the description of a particular command, then that command is unlikely to be of interest to him anyway.

2 Command Structure

Each command has a '#' as its first character. Note that 'hash commands' (as they are known) do not invoke the loader, and there is no one-to-one correspondence between each hash command and an external routine. This means that hash commands cannot (usually) be called from programs, and cannot be aliased.

The parameters to a hash command often include numeric constants. These can be typed as decimal numbers in the normal way, or as hexadecimal numbers, in which case they should be preceded by an 'X'. Commands may be typed in upper or lower case, or a mixture of both. For example, the following two commands would have the same effect:

```
#SWORD(X840000,256)

#sword(x840000,x100)
```

Note that, in all relevant cases, lengths are expressed in bytes.

3 Individual command descriptions

The available commands are as follows.

3.1 #ACR

This command just prints out the current ACR level.

3.2 #CONNECT(filename)

This command connects the specified file. If possible, it is connected in write mode, otherwise read mode is used. The connect address is printed in hexadecimal. filename may describe a member of a partitioned file; in this case the connection mode is always read only. Since a file connected by this command remains connected until explicitly disconnected, is is advisable to do this (using the DISCONNECT command) as soon as any operations on it have been completed. In particular, note that a file connected in write mode will not execute correctly; it must be disconnected first.

3.3 #DEC(value)

This just prints out value in decimal.

3.4 #DUMP(address,length,.out)

This dumps the specified area (in hexadecimal and character form) to out. If out is not specified, .LP is assumed.

3.5 #DUMPFILE(filename,offset of start,length,.out)

This dumps the specified area of the specified file (in hexadecimal and character form) to out. If out is not given, .LP is assumed. The area dumped must lie within the physical limits of the file.

3.6 #HEX(value)

This just prints out value in hexadecimal.

3.7 #LISTFD(channel)

This is used to print out details in the file description record for channel, which must lie in the range 1 to 99.

3.8 #LQUIET

This command will switch off all loader warning messages. This facility is useful if you are PRELOADing files which have a large number of references which have to be made dynamic. In normal circumstances each reference will generate a warning message that its status has changed, and this can generate a lot of non significant warnings. You should only ever use the #LQUIET facility sparingly and in well understood loading situations.

#LQUIET is cancelled by a call on #N.

3.9 #MON

This causes statistics to be printed for each Subsystem command; currently, CPU time and page turns are given. The effect of #MON is turned off by #N (see below).

3.10 #MONITOR

This just executes the IMP statement %MONITOR, and is used mainly for obtaining diagnostic dumps of the Subsystem itself.

3.11 #MONLOAD(mask,out)

This command is used to control the amount of loader diagnostic information generated during loading operations.

It takes two parameters; the first is a bit significant integer value (mask), and the second is the name of a file (out) to which monitoring information is to be sent. Currently the lowest 5 bits of the first parameter are meaningful:

2**0 requests minimal loading information and some important but non-critical warnings.

2**1 requests information on object files which are being loaded and unloaded. Also information on the location and layout of areas in loaded files, and the module source language.

2**2 requests information on names and locations of code and data entry points as they are loaded. Also information on common areas set up by the loader.

2**3 requests information from the loader search module on which entry points are being sought, which directories are being searched, how aliases are handled, etc.

2**4 requests information on which unsatisfied references are being made dynamic when LOADPARM(MIN) is set.

If the second, output, parameter is not specified then diagnostic information will appear on the output terminal (or job journal in the case of a background job), otherwise the parameter should specify an own filename which will be created if it does not exist or overwritten if it does. In this second case, loader monitoring will go directly into the file. Should the file be filled and incapable of further extension then the monitoring will switch automatically to the terminal.

In using #MONLOAD it is generally better to use integer parameters such as 1,3,7,15,31, which have successively more bits set, rather than values such as 2,4,8,16 in which only one bit is set. This is because some information given by higher bits amplifies or expands that given at lower bit setting and the information is no longer seen in context. Note that #MONLOAD(-1) will generate all possible monitoring.

Loader diagnostic monitoring settings will remain in force until another call on #MONLOAD. #MONLOAD(0) or #N will turn off monitoring. Failure messages and some critical warnings are always generated regardless of the #MONLOAD settings (but see the #LQUIET command).

#MONLOAD ? will give the current MONLOAD setting and, if output is being sent to a file, say how much has been written.

3.12 #N

This command just turns off the effect of #MON, #MONLOAD and #SSTOP. It is also useful as a simple method of including comments in an OBEY file, since its effect is usually null (unless other hash commands are being used), and its parameters are ignored. For example:

```
#N This is a comment
```

3.13 #PCOM(n)

n must lie in the range 1 to 60. The contents of location n in COMREG are printed out.

3.14 #PMESS(fault number)

This command prints out the Subsystem error message associated with fault number. Note that any variant information (that usually set by SETFNAME) will not necessarily be correct.

3.15 #PVM

This command prints a map of the user's virtual machine, giving connect addresses and modes, filenames, etc. The output is always to .OUT.

3.16 #QUIT

Sometimes the Subsystem session directory becomes so corrupt (due to a program error) that normal commands (including STOP and QUIT) no longer work. #QUIT will usually enable log-off under these circumstances.

3.17 #REGS(n)

The Subsystem records the contents of the machine registers, and the contingency class and subclass, at each program error. A cyclic buffer of the four most recent sets of values is maintained, and may be accessed by using the #REGS command. #REGS(0) (or just #REGS) prints the most recent set; #REGS(-1) the set before that; and #REGS(-2) and #REGS(-3) the remaining sets.

3.18 #SBYTE(address,value)

This commend is used for altering the value of a single byte in virtual memory; the byte must be connected in write mode. The effect of the command is to set the byte at address to value.

3.19 #SCOM(n,value)

n must lie in the range 1 to 60. Location n in COMREG is set to value.

3.20 #SETBASE(filename)

This command will only work if the user has sufficient privilege. It sets the name of the basefile to be used for subsequent sessions. If the parameter is omitted, the default Subsystem basefile is used.

3.21 #SNAP(address,length)

This command dumps the specified area of virtual memory, in hexadecimal, to .OUT. The amount printed on each line is controlled by the setting of the OPTION ITWIDTH.

3.22 #SNAPCH(address,length)

This command dumps the specified area of virtual memory, as ISO characters, to .OUT. The amount printed on each line is controlled by the setting of the OPTION ITWIDTH.

3.23 #SNAPCODE(address,length,out)

This command decompiles code and sends the output to out. If out is not specified, .OUT is assumed.

3.24 #SSTOP

This command inhibits requests to SPOOLR sent via the SENDFILE routine. The request message that would have been sent to SPOOLR is printed on .OUT; a nonzero flag is returned to the user program. The effect of this command is turned off by the #N command (see above).

3.25 #SSTRING(address,"string")

This command stores string at address. Note that the string should be enclosed in double quote characters.

3.26 #SWORD(address,value)

This commend is used for altering the value of a single word in virtual memory; the word must be connected in write mode. The effect of the command is to set the word at address to value.

4 Error messages

Since the facilities described here are primarily for system programmers, error messages are minimal. The most likely error is a Subsystem fault (a program error in the Subsystem) caused by an invalid address. Subsystem fault 136 may also occur; it is caused by an attempt to write excessive amounts of output (e.g. from #DUMP).

5 Warning

Combinations of the commands described herein are capable of corrupting the contents of any files connected in write mode; the greatest care should be exercised in their use.

6 Acknowledgement

This document is partially based on EMAS 2900 Subsystem Note No. 6, issued on 21st July 1978; it has been considerably updated.

EMAS 2900 SUBSYSTEM NOTE

No:8
Date:20/3/79

Dynamic Loading

This note describes the initial dynamic loading facilities in the Edinburgh Subsystem. For the user of IMP and ALGOL they are very straightforward and operate much as on the 4-75. For the FORTRAN user there is a complication because of the way in which the FORTRAN compiler uses the stack.

Compiling

It is important to appreciate that for a routine to be loaded dynamically it is not necessary to modify the routine in any way. It is the program or procedure that references the routine that must be marked to inform the loader that the reference should be satisfied dynamically. Eventually this will be done at compile time, thus:

ALGOL+FORTRAN: compile with PARM(DYNAMIC) - all references are marked for satisfying dynamically.

IMP: use %DYNAMICROUTINESPEC in place of %EXTERNALROUTINESPEC for all routines and procedures which are to be loaded dynamically. It will also be possible to use PARM(DYNAMIC), as for ALGOL and FORTRAN.

Current compilers do not recognise the dynamic requests, so for a short period it will be necessary to proceed as follows:

Compile the file with any required PARMs.
Mark the file by use of the command DYNAMIC(object file).

To check the references in an object file, use

```
ANALYSE(objectfile, R)
```

Running the program

For IMP and ALGOL there is no further special action required. The program is run or command called just as at present. When the file containing the dynamic references is loaded any references to procedures already loaded are satisfied immediately. All others are left until the first call of the procedure. At this point a jump is made to the loader and an attempt made to load the required procedure. If this fails then messages of the form

```
Failed to load RESET dynamically RESET not found
```

are output, followed by diagnostics and program termination. If it is successful then the reference is filled in the programs GLA (general linkage area) and the routine is called. Subsequent calls behave exactly as for calls to statically loaded routines.

FORTRAN

Where the routines being loaded dynamically include FORTRAN code there is an additional requirement. FORTRAN uses part of the User Stack for its variables, but unlike IMP and ALGOL this space is allocated all the time the file is loaded - not just during the execution of the routine. This so called INITIALISED STACK must be put at the bottom of the user stack, and for this to be possible a space must be allocated. With dynamic loading the loader cannot allocate space because, until execution begins and the additional routines are loaded, it has no information about how much space to leave The method therefore is for the usser to specify how much space is to be allocated for the FORTRAN initialised stack.

This number can be calculated by using the 'S' option in ANALYSE in respect of all the FORTRAN files that are to be loaded in any one RUN. The output includes a table which contains a line of the form

```
ISTK 000038EE 00002AB0
```

The second of these two numbers gives, in hexadecimal, the length of initialised stack for this object file. By adding all of these it is possible to get an accurate figure. This figure should then be specified as the size of the initialised stack using

```
OPTION(INITSTACKSIZE=n)
```

Note that the parameter is in Kilobytes, and that the number specified must be at least 32 (Kilobytes) less than the size of the USERSTACK.

As a simple starting point the following values will probably serve the majority of users:

```
OPTION (USERSTACKSIZE=128)

OPTION (INITSTACKSIZE=60)
```

For those running very large FORTRAN programs or packages, the maximum settings allowed are:

```
OPTION (USERSTACKSIZE=252)

OPTION (INITSTACKSIZE=100)
```

Conclusion

It is hoped that these facilities will prove useful. They may be modified slightly in the light of experience.

R.R. McLeod

Edinburgh 2900 Compilers: Object File Format

This Note describes the format of the object files created by ERCC 2900 compilers using LPUT. It is an intermediate format in the sense that ICL standard object files will be generated from this form, which will not necessarily reside in a permanent file, other than on EMAS. This format is, however, appropriate to the provision of 'fast loaders' in a conventional ICL file system or when object files can be directly mapped into the virtual memory.

The object file may contain up to seven areas in addition to red tape and linkage information, laid out in a contiguous area as below

	area code
standard file header	
code	1
sst	4
plt	3
gla	2
ust	5
initcmn	6
initstack	7
linkage data	
object file map	

The standard file header consists of eight words defined for use on EMAS, and which will be retained on all systems to assist mobility of object code and utilities. These words are used as follows:

word 0 offset of end of data from start of file
1 displacement of start of code from start of file (used by EMAS basic system loaders and compatible with current practice)
2 physical size of the file (bytes)
3 1 (file type for 2900 object)
4 sum check (optional)
5 date and time when file was last written to
6 offset of load data from start of file
7 offset of object file map from start of file

The object file map consists of twenty-two words with the format:

(integer N, recordarray R(1:7))

where

N is the number of areas defined

R has the following format

(integer START, L, PROP)

where R(i) describes area i and
START is the displacement of the start of the area from the start of the object file
L is the length of the area
PROP will be used to define properties of the area (zero at present)

The contents of the seven areas are described in Subsystem Note 15.

LDATA

The linkage data contains a fifteen element array, LDATA, which provides links to records, or lists of records, also held in the linkage data area. These records provide information about entries and references, both for procedures and data items, common areas and relocation requirements between areas. All links within the linkage data are byte displacements from the start of the object file. The significance of the entries in the initial block of pointers is given below. The allocation of these entries has been constrained by the requirement to provide a simple transition from 4-75 EMAS object files and, consequently, some entries are not used.

LDATA	content
0	14 (number of significant pointers following)
1	listhead of procedure entries
2	0
3	0
4	listhead of data entries
5	0
6	0
7	listhead of static procedure references
8	listhead of dynamic procedure references

LDATA	content
9	listhead of data references
10	0
11	0
12	pointer to a record defining object file listing (format to be defined)
13	0
14	listhead of blocks of relocation requests

In the following definitions, identifiers may contain up to 31 characters. However, only sufficient space for the actual identifier (rounded to a word boundary) is reserved. All displacements are in bytes from the start of the appropriate areas, which are identified by the area codes defined above. LINK is the byte displacement of the next record in the list, from the start of the object file map, or zero to terminate the list.

Procedure entries - listhead LDATA(1)

The format of each record is

(integer LINK, LOC, string(31) IDEN)

where

LOC is A<<24!EP and

EP is the displacement of the entry descriptor from the start of area A (2 gla, 3 plt)
IDEN is the name of the entry point

The most significant bit of LOC is set if this is a main program.

Data entries - listhead LDATA(4)

The format of each record is

(integer LINK, DISP, L, A, string(31) IDEN)

where

DISP is the displacement of the data area from the start of area A (2 gla, 5 ust, 6 initcmn)
L is the length of the data area
IDEN is the name of the data area

Procedure references - listheads LDATA(7) and LDATA(8)

The format of each record is

(integer LINK, REFLOC, string(31) IDEN)

where

REFLOC is A<<24!REFAD
REFAD is the displacement from the start of area A (2 gla, 3 plt) of a descriptor which is to be filled in by the loader
IDEN is the name of the referenced procedure

Data references - listhead LDATA(9)

The format of each record is

(integer LINK, REFARRAY, L, string(31) IDEN)

where

REFARRAY is the offset from the start of file of a record of the form

(integer N, integerarray REFLOC(1:N))

where each REFLOC(i) contains A<<24!REFAD and

REFAD is the displacement, from the start of the specified area, of a word which is to have the address of data area IDEN added to it by the loader.
A nominates the area containing the reference (2 gla, 3 plt, 7 initstack).
L is the expected length of the data area

If the reference is to a "common" area, i.e. an area to be created by the loader if not defined, the most significant bit of REFARRAY is also set.

Initialisation data - listhead LDATA(13)

The format of each record is

(integer LINK, A, DISP, LEN, REP, ADDR)

where

A is the area to be initialised
DISP is the offset within the area at which initialisation is to start
LEN is the length of the initialisation data
REP is the number of copies of the initialisation data to be copied to consecutive LEN byte areas
ADDR is the displacement from the start of the file of the initialisation data if LEN >1
it is the initialisation value if LEN =1

Relocation request blocks - listhead LDATA(14)

The format of each record is

(integer LINK, N, recordarray R(1:N))

where

N is the number of relocation entries in this block
R has the following format

(integer AREALOC, BASELOC)

where

AREALOC is AREACODE<<24!AREADISP,
BASELOC is BASECODE<<24!BASEDISP

AREACODE identifies the area containing the item to be relocated.
AREADISP is the byte displacement of aa word within the area to be relocated by the load address of the start of the area identified by
BASECODE which may be any of areas 1 - 7
BASEDISP allows relocation by a sub-component of that area

G.E. Millard

Generation of Object Program Files for 2900

systemroutine LPUT (integer TYPE, P1, P2, P3)

This routine generates an object program file from information passed through the parameters in a sequence of calls. The object file name must previously have been assigned by a string addressed by COMREG(52), and a workfile of at least one segment created and addressed by COMREG(14). (The specification of COMREG is systemintegermap COMREG(integer N).)

The generated object file may have up to seven areas, as defined below, in addition to the linkage information generated by LPUT. The content of each of these areas, any of which may be empty, is at the compiler writer's discretion, subject to the comments below.

area

1 code — should ideally contain only executable code and constants accessed only from within the code area, thus enabling a connect mode of execute, shared. As this mode apparently inhibits the most efficient method of access to constant strings and vectors the connect mode, initially at least, will be execute, read, shared. It is anticipated that connection in execute only mode will be considered essential in some instances.

2 gla (general linkage area) — normally contains descriptors for accessing external objects, entry descriptors and static (normally scalar) data. The connect mode is read, write, unshared. The first eight words have a prescribed use (see below).

3 plt (procedure linkage table) — may be used to contain entry and reference descriptors. If a plt exists then all such descriptors must be contained in it, rather than in gla, which may still exist to contain static data. The connect mode is read, unshared. The first eight words have a prescribed use (see below).

4 sst (shareable symbol tables) — is expected to contain information relating to run-time diagnostics and is connected in read, shared mode.

5 ust (unshared symbol tables) — normally contains static arrays and is connected in read, write, unshared mode.

6 initcmn (initialised common areas) — is an accumulation of separately specified and initialised common areas connected in read, write, unshared mode.

7 initstack (static initialised area on stack) — may be used to contain local data and data descriptors. This area should not normally be used for arrays as total stack space is constrained.

The first eight words of the gla, or plt if it exists, are currently defined as follows:

```
word 0     code descriptor to the
     1     first or only entry point
     2     address of ust area
     3     address of sst area
     4     byte 0 language flag 1 IMP
                                2 FORTE
                                3 IMPS
                                4 NASS
                                5 ALGOL
                                6 optimised code (no diag tables)
                                7 PASCAL
                                8 SIMULA
           byte 1 compiler version
           byte 2 compiler options
           byte 3 (may be language dependent)
     5     reserved for address of gla if plt is used, otherwise 0
     6     reserved
     7     reserved
```

After opening the object file (TYPE=0) fragments for any of the areas may be passed to LPUT in any order. The location for the fragment within the object file is specified relative to the relevant base. If a fragment overlaps a previous fragment then that part of the previous fragment is replaced. Additional calls on LPUT can provide information to enable linkage to other external objects to be achieved and also to specify locations in the non-shareable areas which are to be relocated by the actual value of one of the bases when the object file is loaded. When all the information relating to the current external routine has been passed to LPUT a type 6 call confirms the sizes of 1 to 5 (the sizes of initialised common areas having been separately specified).

Subsequent calls on LPUT will generate a new set of areas. A type 7 call indicates that the file is now complete. The type 6 call may be omitted if only one set of areas is produced.

The generated object file has, in its simplest form, the layout shown below:

```
header
code
sst
[plt]
gla
ust
initcmn
initstack
linkage data
```

A particular compiler may choose to generate only one set of areas even when compiling a number of external routines. Alternatively each external routine may generate a separate set of areas. The layout of an object file with multiple sets of areas is shown in the figure below. (This also represents the layout of the file generated by linking files generated by separate compilations).

```
header
code 1
code 2
sst 1
sst 2
[plt1]
[plt2]
gla 1
gla 2
ust 1
ust 2
initcmn 1
initcmn 2
initstack 1
initstack 2
linkage data
```

No assumptions should be made about the relative position of the areas when a program is loaded. An auxiliary stack (for use by dynamic data structures) may be located by a data reference to 'ICL9CEAUXST'. This is a two word location containing a descriptor to an area with the format below:

```
word 0     current 'free aux stack' pointer (absolute)
     1     reserved
     2     physical end of aux stack
     3     reserved
```

Calls on LPUT

The calls on LPUT are as follows. Any parameters not specified for a particular call should be set to zero to allow for future expansion. All displacements must be in bytes.

TYPE =0 Initialisation call. The object file is opened.
P1 = language flag
P2 = release no.
P3 = version no.

TYPE =1 code area
2 gla area
3 plt area
4 sst area
5 ust area
This set of calls specify that P1 bytes of information (a 'fragment') currently held at address P3 in the calling routine are to be placed at the location P2 bytes from the start of the nominated area, unless P3 < 256. In this case P1 bytes each containing the value P3 are to be set up at the location specified by P2.

TYPE =31 code area
32 gla area
33 plt area
34 sst area
35 ust area
36 initcmn area
37 initstack area

This set of calls act as for 1 - 5 above, but providing for the extended set of areas.

TYPE =41 code area
42 gla area
43 plt area
44 sst area
45 ust area
46 initcmn area
47 initstack area
This set of calls provides for repeated initialisation with up to 255 bytes of data.
P1 = number of bytes of data <<24! number of copies
P2 = displacement from area start at which initialisation is TO commence
P3 = address of the initialising data

TYPE =6 This call indicates that the information relating to the current set of areas is complete.
P1 = 32 (current length of information)
P3 = address of eight word information field containing the lengths of areas 1 to 7 and the sum of these lengths for the preceding set of areas.

TYPE =7 Indicates that the information relating to the current file is complete.
P1 = 32
P3 = similar to type 6, but with sub-totals for each area type and the grand total.

TYPE =10 Define a 'common' area reference
P1, P2 and P3 are as for type 15, substituting 'common' for data.

TYPE =11 Define an entry point.
P1 = code of the area containing the entry descriptor (2 gla, 3 plt). If the most significant bit is set this signifies that the entry point is to a main program unit.
P2 = location of the entry descriptor relative to the start of the area defined by P1.
P3 = address of a string defining the entry point name. This is truncated to the most significant 31 characters.

TYPE =12 Define an external routine reference.
P1 = code of the area containing the reference block (2 gla, 3 plt).
P2 = location of a two-word reference block relative to the start of the area defined by P1. This will be filled, at load time, with a descriptor through which a call may be made to the referenced routine.
P3 = address of a string defining the name of the referenced routine. (31 characters significant).

TYPE =13 Define an external routine reference which is to be satisfied dynamically, i.e. when it is actually called. If dynamic loading is not available this is treated as type 12.
P2 and P3 are as for type 12.

TYPE =14 Define a data entry point.
- P1 = (AREA <<24) ! length of the data area, where AREA identifies the area containing the data area (2 gla, 5 ust, 6 initcmn). If AREA is 6 this is an initialised common area.
- P2 = location of the data area relative to the start of area AREA.
- P3 = address of a string defining the data area name (31 characters significant).

TYPE =15 Define a data area reference.
- P1 = (AREA CODE <<24)! minimum length the data area is expected to have, where AREA CODE identifies the area containing the reference (2 gla, 3 plt, 7 initstack).
- P2 = location of a word relative to the start of the area defined by AREA CODE which will have the address of the data area added to it when the object file is loaded.
- P3 = address of a string giving the name of the referenced data area.

TYPE =17 Define a data entry point in unshared symbol tables (area 5).
- P1 = length of the data area.
- P2 = locations of the data area relative to the start of gla symbol tables.
- P3 = address of a string (maximum 31 characters) defining the data area name.

TYPE =18 Relocate an 18 ~~last~~ bit address in code (area 1).
- P1 = 0
- P2 = displacement from the start of the code area of a 4 byte instruction.
- P3 = 18 bit address to be added to the last 18 bits of the instruction located by P2.

TYPE =19 Request relocation of a single word.
- P1 = code for area containing the reference:
  - 2 gla
  - 3 plt
  - 5 ust
  - 7 initstack
- P2 = byte displacement within area P1 of a word to be relocated at load time by the base of the area specified by P3.
- P3 = code for area base:
  - 1 code
  - 2 gla
  - 3 plt
  - 4 sst
  - 5 ust
  - 7 initstack

G.E. Millard

No: 16
Date: 31/8/79

Use of COMREG for Storing PARM Options

The PARM options used by the compilers, by LINK and by the Loader are held in elements 27 and 28 of COMREG. COMREG must be specified if access is required to it:

systemintegermapspec COMREG(integer N)

The table below associates the pairs of PARM options with the bits of COMREG(27) and COMREG(28). The effects of these PARMs are described in Subsystem Note 7.

COMREG(27)

Bit n (2**n)	Set (Default underlined)	Unset
31	n/a	
30	JOBBER MODE	NORMAL MODE
	(set by Jobber)	
29	MISMATCH	NOMISMATCH
28	PARMX	NOPARMX
27	PARMY	NOPARMY
26	PARMZ	NOPARMZ
25	spare	
24	Stack size set	
23	NOLINE	LINE
22	EBCDIC	ISO
21	Diag stream set	
20	DYNAMIC	STATIC
19	FREE	FIXED
18	DEBUG	NODEBUG
17	MAP	NOMAP
16	OPT	NOOPT
15	ATTR	NOATTR
14	CODE	NOCODE
13	LET	NOLET
12	LABELS	NOLABELS
11	XREF	NOXREF
10	ZERO	NOZERO
9	INHIBIOF	ALLOWIOF
8	IMPS	IMP
	(compiler command)	
7	PROFILE	NOPROFILE
6	NOTRACE	TRACE
5	NOARRAY	ARRAY
4	NOCHECK	CHECK
3	STACK	NOSTACK
2	NODIAG	DIAG
1	NOLIST	LIST
0	QUOTES	PERCENT

COMREG(28)

Bit n (2**n)	Set (Default underlined)	Unset
31-16	n/a	
15	MINSTACK	NORMALSTACK
14	(May be used for more 'stack' options)	
13-9	n/a	
3	R8	R4
2	L8	L4
1	I8	I4
0	n/a	

J.M. Murison

Modifying Object Files on EMAS 2900

This note describes a program which performs a variety of operations on object files. It is likely to be useful where large object files with many entries and references are handled.

The following types of operation are available:

- changing the names of procedure or data entries and references.
- suppressing procedure and data entries.
- changing static procedure references to dynamic and vice versa.
- merging the code and shareable symbol tables areas or the GLA and unshared symbol tables areas.
- binding the object file.

The BIND operation is intended to alleviate the problem of lengthy loading times for large packages. Essentially, it processes an object file to produce a module which can subsequently be loaded at minimal cost. To do this it assumes fixed sites for the code, GLA and stack, and hence can satisfy all relocation requests. It also resolves all cross-references in the object file.

The program is accessed by:

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)
```

and called by:

```
Command:MODIFY(object file [,output object file] [,report file])
```

24/4/81 | The input (and output) object file may be a pd file member.

If the second parameter is omitted, the modified object file overwrites the input object file.

If the third parameter is omitted, the report, which contains details of the operations performed, is sent to T#MODLIST. Fault messages are sent to the terminal and do not appear in the report.

The program issues the prompt "Operation:", the replies to which are given below. Most operations require further input and issue appropriate prompts.

The "Operation:" prompt is reissued until the reply "CLOSE" is given. This terminates the modification and causes the output file to be generated.

All input may be given in upper or lower case; spaces are not significant.

The parameters for the first five operations consist of one or more lines, each containing a pair of names separated by a comma, the list being terminated by .END, e.g.

```
Operation:RENAME
Proc ent pair:DREAD, READDATA
Proc ent pair:DPRINT, PRINTDATA
Proc ent pair:.END
Operation:
```

1. RENAME

   Changes the names of procedure entries.

   Parameters: procedure entry, newname

2. REDIRECT

   Changes the names of procedure references.

   Parameters: procedure reference, newname

3. RENAME DATA

   Changes the names of data entries.

   Parameters: data entry, newname

4. REDIRECT DATA

   Changes the names of data references.

   Parameters: data reference, newname

5. ALIAS

   Allows a copy of a procedure entry to be made and given a different name.

   Parameters: procedure entry, copy name

The parameters for operations 7-13 below consist of a list of items separated by commas, or the keyword .ALL. The list is terminated by newline unless the last character is a comma, in which case it continues on the following line. The remaining operations (14-17) take no parameters.

Examples:

```
Operation:MAKEDYNAMIC
Proc ref list: A1,A2,A3,
Proc ref list: A4
Operation:
```

```
Operation:SUPPRESS DATA
Proc ent list:.ALL
Operation:BIND
Operation:
```

6. MAKE DYNAMIC

   Makes static procedure references dynamic.

   Parameters: static procedure reference list or .ALL

7. MAKE STATIC

Makes dynamic procedure references static.

Parameters: dynamic procedure reference list or .ALL

8. SUPPRESS

This operation allows procedure entries to be suppressed. Other operations such as SATISFY REFS will still be able to find and use a suppressed entry - the only effect is that the suppressed entry will not appear in the load data for the output object file. Note that a main program entry cannot be suppressed.

Parameters: procedure entry list or .ALL

9. RETAIN

This cancels a SUPPRESS request. Since by default all entries are retained, this operation is provided so that SUPPRESS with parameter .ALL can be followed by explicit RETAIN calls for the procedure entries which are actually required. Note that an object file must have at least one main entry, procedure entry or data entry.

Parameters: procedure entry list or .ALL

10. SUPPRESS DATA

This command allows data entries to be suppressed (see SUPPRESS).

Parameters: data entry list or .ALL

11. RETAIN DATA

This cancels a SUPPRESS DATA request (see RETAIN).

Parameters: data entry list or .ALL

12. SATISFY REFS

For each procedure reference given, a search is made of the list of procedure entries. If a matching entry is found, then the reference is satisfied and removed from the load data. This operation is performed as a side effect of BIND, and is hence redundant if BIND is also called.

Parameters: external procedure reference list or .ALL

13. SATISFY DATA

For each data reference given, a search is made of the list of data entries. If a matching data entry is found, then the data reference is satisfied and removed from the load data. This operation is performed as a side effect of BIND, and is hence redundant if BIND is also called.

Parameters: data reference list or .ALL

14. FUSE CODE

This causes the shareable symbol tables (SST) to be permanently appended to the code area. Hence a subsequent LINK command will not have its usual effect of separating these areas - LINK normally collects all code areas together and follows this by a corresponding series of SSTs. The intention is to prevent a degradation of paging behaviour. All relocation requests with base SST are amended to use the code area as base address.

Parameters: None

15. FUSE GLA

This causes the unshared symbol tables (UST) to be permanently appended to the GLA. See FUSE CODE.

All relocation requests with base UST are amended to use the GLA area as base address. In addition, the list of data entries is searched. If any is defined with respect to the UST, then the definition is amended to refer to the GLA.

Parameters: None

16. BIND

This command performs pre-loading operations on the object file so that it can be loaded subsequently at reduced cost. To do this it assumes fixed sites for the code, GLA and stack, and hence can satisfy all relocation requests. It also creates an area for uninitialised COMMON and satisfies data references to it. An entry is then added to the object file listing. To achieve the greatest savings in load time, the BIND command should be preceded by calls on the commands to suppress procedure and data entries. BIND automatically attempts to satisfy all procedure and data cross-references, so the explicit commands for these operations need not be called where BIND is used.

Note that once bound, an object file cannot subsequently be modified or linked. In addition, a bound file must be the first file loaded if called at all; this also means that only a single bound file may be loaded at a time.

Parameters: None

17. CLOSE

This terminates the user input. A confirmatory message is printed if the modification is completed successfully.

Sandy Shaw

Use of COMREG (27) for Compile Time Options

The following table shows the current use of the bits in COMREG (27). COMREG (26) is also allocated for use for additional PARMs.

Bit n (2 ** n)	SET	UNSET	NOTES
31			Not yet allocated
30	JOBBER MODE	NORMAL MODE	Not set by user
29	MISMATCH	NO MISMATCH	Fortran only
28	R8/PARMX	R4	Fortran - default REAL*8. Imp for test facility.
27	L8/PARMY	L4	Fortran - default LOGICAL*8. Imp for test facility.
26	I8/PARMZ	I4	Fortran - default INTEGER*8. Imp for test facility.
25			Not yet allocated.
24	Stack Size Set		Temporary use - not set by user
23	NOLINE	LINE	
22	EBCDIC	ISO	
21	Diag stream set		Temporary use - not set by user
20	DYNAMIC		
19	FREE	FIXED	
18	DEBUG		
17	MAP		Used by command LINK
16	OPT		
15	ATTR		Fortran - list attributes of variables
14	CODE		
13	LET		
12	LABELS	NO LABELS	
11	XREF	NO XREF	Fortran - list variable cross reference list.
10	ZERO		
9	INHIBIOF		Fortran - inhibit some I/O checks
8	IMPS	IMP	
7	PROFILE		
6	NOTRACE	TRACE	
5	NOARRAY	ARRAY	
4	NOCHECK	CHECK	
3	STACK		
2	NODIAG	DIAG	
1	NOLIST		
0	QUOTES	PERCENT	

Roderick McLeod, October 1978

This command allows EMAS users on machines attached to the RCOnet to send each other messages. The current pre-release version operates only on the 2972.
Please send suggestions or comments to S. Shaw.
For the present, INSERT(ERCC27.MAILY) to gain access to the command.

Basic operations

MAIL provides a number of facilities for manipulating messages; however to get started, the subset described below should be sufficient.

1. Composing and sending a message

First invoke MAIL:

```
Command:MAIL
Mail:
```

The prompt 'Mail:' is issued whenever MAIL expects the next command to be specified. To send a message to "Bill Smith", first check on the correct form of the name to use, (it may be "B.Smith" or "W.Smith" or "B.A.Smith" etc) :

```
Mail:DIRECTORY *SMITH
B.Smith, H.T.Smith, S.Smith
```

Then use COMPOSE to create the draft message:

```
Mail:COMPOSE
Subj: Test Message
To:   B.Smith
Text:
: Here is a test message
::
Send or send-and-file-a-copy now?
Y or Fcc or N:
```

If you are happy with the message, send it by replying "Y" ("Fcc" will in addition file a copy of it). If the reply "N" is given, then the contents of the draft can be amended or reviewed before sending:

```
Mail:LIST DRAFT
```

Once the draft is satisfactory, use SEND to dispatch it:

```
Mail:SEND
Message sent
```

The draft message may be filed without being sent:

```
Mail:FILE DRAFT
```

2. Receiving and displaying messages

You can receive messages into your process either when calling MAIL or by subsequent use of the ACCEPT command:

```
Command:MAIL(*)
```

If there are any outstanding messages, they are taken from MAILER and a one line summary is printed for each. To accept messages when you are already within MAIL:

```
Mail:ACCEPT
```

Newly received messages can be listed on the console:

```
Mail:LIST NEW
```

No action is required to retain received messages; see DISCARD, described below, about deleting them.

3. Reviewing and deleting mesages

A one line summary of each message held can be produced using SCAN:

```
Mail:SCAN ALL
```

The sequence number printed at the start of each one line entry can be used as a parameter to LIST or DISCARD, described below.

To review specific messages, find the required sequence number from the SCAN output, then use it with LIST; e.g., for message 3:

```
Mail:LIST 3
```

To get rid of unwanted messages use DISCARD:

```
Mail:DISCARD 3
```

To get rid of all currently held messages use the keyword ALL:

```
Mail:DISCARD ALL
```

Discarding simply marks messages as eligible for removal. A subsequent call of TIDY is needed to purge them entirely:

```
Mail:TIDY
```

On-line assistance is provided by HELP. Given no parameter it offers a table of contents; otherwise it lists information about a specified MAIL command. Return from viewing the help file back to MAIL by typing QUIT:

```
Mail:HELP SCAN
.
.
.
View:QUIT
```

Overview

The MAIL command provides facilities for composing, amending, sending, receiving and storing messages. Messages are held in store map files called folders. All the correspondence which relates to one topic can be conveniently held in one folder.

In turn, messages themselves consist of a number of components. The body of the message consists of a text component, and the header of the message consists of all the other components, e.g.

```
Subj: Meeting on Wednesday, 2 p.m.
From: S Shaw
To:   J Smith, j jones

Please note the new location for our meeting is Room 2019.
S.
```

The process of composing a message entails placing text into the various components of the draft message (which is not held in a folder). For example, addresses go into the "To:" and "cc:" components and the body of the message goes into the "Text:" component. A draft message can be Sent which causes its delivery to all the indicated recipients; each will receive a one-line TELL message indicating that they have outstanding mail. After Accepting messages, the user may selectively List them. Further manipulation of a message can involve Forwarding copies to additional recipients, Replying to its author, Filing for later reference or Discarding.

The messages in a folder are ordered chronologically and may be referenced by their index number (i.e. by their position in the folder), or by using special labels. At any given moment the system has a current folder and within it a current message which are under scrutiny; this avoids having to specify a folder and message index for every command.

A standard folder called M#INBOX is created by the system. When the MAIL command is invoked, the normal action is to Open the standard folder and select it as the current folder.

Only the draft message may be modified. However, any message in a folder can be Copied to the draft; similarly, the draft message can be Filed in a folder.

You are notified when a message has arrived for you by a TELL message from the executive process MAILER. At this stage the message has not been placed in your process - you must call the MAIL command and explicitly Accept it to cause the transfer to be performed.

Calling MAIL

The MAIL command takes two parameters, both optional:

```
Command:MAIL(<R-name>/<folder name>)
```

With no parameters, the program issues the prompt 'Mail:' to indicate that it is expecting a MAIL directive to be given. This prompt is

reissued after each directive has been performed until the directive STOP or QUIT is given:

```
Command:MAIL
Mail:
```

If a folder name is specified (preceded by '/'), that folder is OPENed and hence made the current folder; otherwise M#INBOX is made the current folder:

```
Command:MAIL(/RFOLDER)
Folder RFOLDER contains 16 messages
```

If <R-name> is specified (or '*', interpreted as the surname string of the process in use) then an ACCEPT <R-name> command is invoked, i.e. mail from other users is accepted:

```
Command:MAIL(*)
2 new messages
.
.
Mail:
```

The parameters for each MAIL command generally take the form <input>/<output>. If no <output> parameter is given then the slash (/) can be omitted.

A command name need not be typed out in full and in most cases may be abbreviated to a single letter. Where an ambiguous name is given, e.g. "F", then alphabetical order determines the command selected. Hence "F" will invoke "File", "FO" will invoke "Forward". Lower case input is accepted throughout.

Summary of commands

```
Accept    - take outstanding messages; may be implied when calling MAIL
Accredit  - add an alias R-name to the name/address directory
Compose   - create a draft message and offer to SEND it
Copy      - create (or append to) components of the draft message
Directory - list an extract of the name/address directory
Discard   - mark messages as discarded, or destroy draft components
Discredit - remove an R-name from the name/address directory
Ecce      - invoke ECCE to edit a component of the draft message
Edit      - invoke the standard editor to edit a component of the draft
File      - copy messages to another folder, then discard the originals
Forward   - package up an existing message for retransmission
Goto      - make the message specified the current message
Help      - provide on-line information about MAIL commands
List      - display messages on the console or list to a file
Next      - list the next message in the folder on the console
Open      - make the folder named the new current folder
Output    - list a message component to a file,device or the console
Previous  - list the previous message on the console
Quit      - exit from MAIL and return to the Subsystem
Reply     - create a draft message in reply to one received
Retrieve  - remove "discarded" status from messages
Scan      - produce a list of contents for the current folder
Send      - package up a message and submit it for transmission
Stop      - exit from MAIL and return to the Subsystem
Tidy      - purge discarded messages from a folder
```

Addressing conventions

Messages are addressed to individuals by using their "recipient names" (R-names). Each R-name is unique and, for EMAS users, corresponds to the surname string associated with their EMAS process (the same string is printed on line printer output banners). Every EMAS user with a unique R-name is automatically accredited to the MAIL system, i.e. the name is entered in MAIL's name/address directory. Information on R-names known to MAIL can be found by means of the command DIRECTORY (see below). A user can have his surname string (and hence R-name) changed by making a request to the System Manager.

Each R-name has an associated address of the form [username @ host]. Either the R-name or the address may be used to direct messages, e.g.

```
S.Shaw   and   [ERCC27 @ 2972]
```

are both valid forms for specifying recipients.

When a specified R-name is being processed by MAIL, the case of the letters is ignored, as are spaces and dots.

A user can avoid having his surname string made known to other users as an R-name by setting permission NONE to the executive process MAILER:

```
Command: PERMIT(.ALL,MAILER,N)
```

An arbitrary number of R-names may be associated with a given address. This may be convenient where several people share a process or where one person has several roles. Requesting additional R-names as aliases is provided as a user facility (see the command "Accredit", below).

Message lists

Within a folder, messages can be referenced by index number (position in the folder), and a collection of messages can be referenced at one time, by using commas and dashes as connectors. Hence the specification "1, 7-9, 90>89" refers to messages one, seven, eight, nine, ninety and eighty-nine. The angle bracket is like dash except that it indicates that the sub-list is in descending order.

Also, certain keywords define groups of messages, so that "new, 10-15, last" will reference all new messages, the tenth to fifteenth and the last message in the folder. The same message may appear more than once in such a list, but this does not imply any "repetition" of the message.

Keywords may be abbreviated to a shorter form (as short as a single letter). But note that "D" is a short form of "draft" (not "discarded") and "N" is a short form of "new" (not "next").

The terms defined below (with the exception of "draft") relate to messages in the current folder; they are as follows:

n	-	message n in the folder
n1-n2	-	messages n1 to n2 (n1 less than n2)
n2>n1	-	messages n2 to n1 (n2 greater than n1)

all	-	all messages in the folder
current	-	the message currently under scrutiny
discarded	-	messages which have been discarded to a "wastebin" but which the janitor has not yet taken away (see the "Discard" command, below)
draft	-	the single message which is currently being prepared; it is not held in any folder
last	-	the last message in the folder
new	-	received messages which have not yet been LISTed
next	-	the first message after the current one
old	-	messages in the folder which are not discarded, new or saved
previous	-	the first message preceding the current one
saved	-	former draft messages which have been saved by being FILEd in the folder

Message components

A message is composed of a series of components. The body of the message consists of a "Text:" component - all the other components together constitute the header of the message. The following components are defined:

Date: Indicates the date and time when the message was sent.

Subj: Gives a brief indication of the content of the message and is displayed when the message is SCANned.

From: Indicates who sent the message. This field may be set by the sender to contain any text he wishes (if, for example, he is sending a message on someone else's behalf); in this case, MAILER will add a "Sender:" component to the message to show who actually sent it.

Sender: Shows who actually sent the message and indicates that the "From:" component is not authentic.

To: The one or more primary recipients of the message.

cc: One or more secondary recipients (who receive carbon copies).

bcc: One or more tertiary recipients (who receive blind carbon copies, see below).

Msg ID: This holds a unique message identifier and is composed of three parts:

1) message server name (= host)
2) a numeric identifier allocated by the server
3) date and time when the message was sent

In reply to: This component is added to a message by the REPLY command, and identifies the message being replied to.

References: Not filled in or used by any MAIL function, this component may contain any text.

Keyword: Again, this component may contain any text and may be used to classify messages or to direct the operations of programs which automatically receive and manipulate messages.

Folder: Indicates the name of a folder in which the sender recommends the recipient to file the message.

Mode: If specified, may contain the value BINARY or EHEADER. BINARY indicates that the body of the message is a binary file. EHEADER indicates that the body of the message includes a standard EMAS header.
The LIST command does not attempt to display the body of a message which has either mode set. The original file may be reconstituted using the OUTPUT command.

When sending a message to a number of recipients, the sender may determine how much information each recipient receives about his co-recipients by selective use of the "To:", "cc:, and "bcc:" components.

To: a recipient in this list has details of all the recipients included in his copy of the message

cc: a "cc:" recipient does not have the "bcc:" component included in his copy of the message.

bcc: a "bcc:" recipient will receive a copy of the message containing neither the "cc:" nor "bcc:" components; his own name is added to the "To:" component in his copy of the message.

Hence, a message addressed

```
To:     Black
cc:     Brown
bcc:    White
```

will be received by each as follows:

```
Black - To:   Black
        cc:   Brown
        bcc:  White

Brown - To:   Black
        cc:   Brown

White - To:   Black, White
```

This feature may be useful where a distribution list containing many names is given - the sender can avoid burdening each recipient with a long list of names in which he has no interest.

Many of the commands of the MAIL program take a message component parameter. The full specification is:

<component name>:<message>(<folder name>)

e.g. cc:4(LETTERS)

The folder name can usually be omitted - by default the message referred to will be in the current folder. The <message> part of the specification may also be omitted, in which case the current message under scrutiny is assumed. Hence the message component specification "To:", refers to the "To:" component of the current message in the current folder.

Message component names can be abbreviated to a shorter form - in all cases, the first two characters of each component name give a unique abbreviation (hence "SE:" is equivalent to "SENDER:").

Note that a message keyword which may define more than one message can be used - the message selected is the first message found which matches the keyword.
Hence "Subj:NEW" refers to the "Subj:" component of the first NEW message in the current folder.

Mail commands

Mail:ACCEPT <R-name>, <password> / <folder name>

This command is used to accept messages sent to you by other users. By default, the messages are put in the current folder:

Mail:ACCEPT

If the <folder name> parameter is specified, then the messages are stored in that folder (it is made the current folder). The <R-name> parameter is required in two cases:

- where you want to accept mail directed to an alias R-name, e.g.

  Mail:ACCEPT EMAS Suggestions/SUGGBOX (puts mail addressed to "EMAS Suggestions" into folder SUGGBOX)

- where you are accepting mail within an EMAS process other than your own, e.g.

  Mail:ACCEPT S Shaw, PASS

After taking the outstanding messages, a SCAN of new messages (i.e. messages not yet LISTed) is performed. The first new message in the folder becomes the current message.

Mail:ACCREDIT

This command allows users to add additional R-names for themselves (aliases) to the name/address directory. This may be useful where several people share one EMAS process, or where one person has several roles. The facility should not be used to define names which

will be meaningless to most MAIL users, such as nicknames known to only one or two people.

You may also set a password for the R-name which will make it possible to accept messages at another address (see ACCEPT). Passwords may be up to seven characters long.

It is also possible to set a "Department" field (31 characters) to be associated with the R-name. The information is displayed when a search is made of the name/address directory (see DIRECTORY) and is intended as an aid to distinguishing recipients with similar R-names.

Note that if you want to have your default R-name changed, this can only be done by application to the System Manager.

```
Mail:ACCREDIT
Rname:      Mail Suggestions
Password:   ABCDE
Department: EMAS MAIL Suggestion Box
```

Mail:COMPOSE <component names>

This command offers a convenient way of creating a draft message. The draft is first cleared, then prompts are issued as indicated:

```
Mail:COMPOSE
To: J.Jones
Subj: Old joke
Text:
: Send 3/4d were going to a dance.
::
Send or send-and-file-a-copy now?
Y or Fcc or N:Y
Message sent
```

Once the draft has been composed, the program offers the option of sending the message or returning to MAIL command level. Input for each component (except "Text:") is terminated by a newline unless the character before the newline is a comma; thus, for example, the "To:" component can be given a list of R-names which extends over several lines. Null input is accepted.
The "Text:" component is terminated by a colon (:) on a line by itself. The prompt printed when lines of the "Text:" component are being input is ':'.

In addition to accepting text, the COMPOSE command will accept the contents of an EMAS file or of a component of an existing message. The escape character '@' must be used to indicate this form of input. You may request that prompts are issued for additional message components by giving the component names as parameters:

```
Mail:COMPOSE cc:
To: @ERCC27.NAMES
cc: J.Jones
Subj: @Subj:
Text:
:@text:4
Send or send-and-file-a-copy now?
Y or Fcc or N: N
```

In this example, a file is copied to the "To:" component of the draft, the "Subj:" component of the current message is copied to the "Subj:" component of the draft, and the "Text:" component of message 4 is copied to the "Text:" component of the draft.

If the draft message is not sent (as in the example above), it may be modified further then dispatched using the SEND command.

Mail:COPY <input>/<component of the draft or DRAFT>

This command allows text to be copied to a component of the draft. Alternatively a complete message may be copied to the draft. If <input> is not specified then a prompt is issued and the text to be copied is read from the terminal. If the component of the draft is not specified, the "Text:" component is assumed. Alternative sources of input are an EMAS file or a component of an existing message.

```
Mail:COPY              (No parameters, so input is prompted
Text:                  for and is taken to be for the "Text:"
:This is input         component of the draft message)
::

Mail:COPY MSGTEXT      (EMAS file MSGTEXT is copied to the "Text:"
                       component of the draft message)

Mail:COPY CC:/TO:      (copies the "cc:" component of the current
                       message to the "To:" component of the draft
                       message)
```

If the input is to be appended to the existing contents of a component of the draft message rather than overwrite it then the symbol '+' should be inserted before the component name:

```
Mail:COPY MYLIST/+TO:  (appends the contents of the file MYLIST to
                       the "To:" component of the draft)

Mail:COPY CC:4/+CC:    (appends the contents of the "cc:" component
                       of message 4 in the current folder to the
                       "cc:" component of the draft)
```

Alternatively, a complete message (i.e. all its components) may be copied to the draft message. In this case the default for the first parameter is the current message.

```
Mail:COPY /DRAFT       (copies the current message to the draft)

Mail:COPY NEW/DRAFT    (copies the first new message in the
                       current folder to the draft)
```

Mail:DIRECTORY <R-name mask>, FULL/<output>

This command allows a search to be made of the name/address directory. The search may be for a specific R-name, or for all R-names that fit a mask. As in the Subsystem command FILES, the mask consists of up to three fields where a field is either a string of explicit characters or the symbol "*", representing any characters. Upper and lower case characters are not distinguished, and space and dot characters are ignored.

If the second parameter ("FULL") is specified, then additional information is given on each R-name selected, under the following headings:

User - the user number to which messages are delivered

Server - the host on which the recipient is accredited

Options - currently takes one of two values:

S'name - the R-name is the standard process surname string, set by the System Manager

Alias - the R-name was accredited by the user himself

Dept - a user-defined field set by ACCREDIT

The <output> parameter may be null (implying output to the terminal), or a filename or device name.

```
Mail:DIRECTORY *shaw
A.Shaw, B.C.  Shaw

Mail:DIRECTORY *MAC*/.LP

Mail:DIRECTORY *JONES,FULL/TEMP1
```

Mail:DISCARD <messages or draft components>

This command marks one or more messages in the current folder as being discarded, but does not physically remove them or re-number the remaining messages in the folder. The action is like placing a message in the wastebin - it is still available though less convenient to access, and is subject to permanent removal later by the TIDY command (see below). If no TIDY has been performed after a DISCARD then the RETRIEVE command can be used to recover the messages.

As with other commands, DISCARD can be applied to the draft. In addition, for the draft only, individual components may be discarded. (However, discarded components of the draft are destroyed immediately and cannot be recovered by RETRIEVE.)

```
Mail:DISCARD 1-4,SAVED,CC:  (discards messages 1-4 and all SAVED
                            messages in the current folder, plus the
                            "cc:" component of the draft)

Mail:DISCARD DRAFT          (discards all components of the draft
                            message)
```

The last message specified in the list becomes the current message if it is not the draft.

Mail:DISCREDIT

This command removes an R-name from the name/address directory. A password must be supplied if the command is called from any process other than that associated with the R-name. In order to have your default R-name removed permanently from the name/address directory it is necessary to set a file index permission of "none" to MAILER:

```
Command: PERMIT(.ALL,MAILER,N)
```

Discredit is called as follows:

```
Mail:DISCREDIT
Rname: MAIL Suggestions
Password: ABCDE
```

Mail:ECCE <component or EMAS file>/<component of the draft>
Mail:EDIT <component or EMAS file>/<component of the draft>

This command allows an EMAS file or an existing component of any message file to be edited, and the result placed in a component of the draft.

If no parameters are given, then the "Text:" component of the draft message is edited. If an output draft component (i.e. one following "/") is given, then the existing contents of that component are overwritten.

Mail:ECCE	- edits the "Text:" component of the draft, creating it if none already exists
Mail:EDIT CC:	- edits the "cc:" component of the draft, creating it if none exists.
Mail:ECCE Text:current/text:	- edits the "Text:" component of the current message to the "Text:" component of the draft.
Mail:EDIT /TO:	- edits an empty file to the "To:" component of the draft.
Mail:EDIT MYLIST_N1/TO:	- edits an EMAS file to the "To:" component of the draft.
Mail:ECCE CC:2	- this will fail; a component of the draft must be given following "/" in this case.

Mail:FILE <list of messages in the current folder>/<folder>

This command copies messages from the current folder to another folder, then discards the messages from the current folder. The input list of messages defaults to the current message. The draft message may also be filed (this is the only message which can be filed in the current folder); a filed draft message is given SAVED status (see SCAN).

The last message in the list of messages to be filed becomes the current message.

```
Mail:FILE /folder2      - files the current message to folder2

Mail:FILE DRAFT         - files the draft to the current folder

Mail:FILE NEW,1,LAST/folderb  - files all new messages plus the
                                first and last in the folder to folderb
```

Mail:FORWARD <message>

This command sends a copy of a message to another user or users. MAIL issues a prompt for the name of the user or users to whom you wish to forward the message. You reply to this with the name(s) or alternatively specify an EMAS file or a message component which contains the names.

It is often useful to add comments to the end of a forwarded message. This can be done after FORWARDing but before SENDing the message by using COPY or EDIT.

As with COMPOSE and REPLY, you are offered the option of sending the message once the recipients have been specified. If <message> lies within the current folder, it becomes the current message.

```
Mail:FORWARD 1          - forward the first message in the
To: Rowland Hill          current folder

Mail:FORWARD LAST(FOLDER2)  - forward the last message in folder2
To: @MYGROUP                to the list of recipients specified
                            in file MYGROUP

Mail:FORWARD NEW        - forward the first new message in the
To: @cc:                  current folder to the list of recipients
                          held in the "cc:" component of the current
                          message.
```

Mail:GOTO <Message>

The message specified becomes the current message. If a message keyword is used which may select more than one message, the first message found is selected.

```
Mail:GOTO 1     - go to the first message in the current folder

Mail:GOTO NEXT  - the next message after the current one

Mail:GOTO NEW,LAST  - go to the first new message; if there are
                      none, go to the last message in the current
                      folder
```

Mail:HELP <command>

This command provides information about the MAIL system and includes descriptions of all the MAIL commands. It operates by VIEWing a file containing the help text; hence the whole of this file can be explored at one time. If no parameter is given a table of contents is printed and further input requested. Alternatively, the name of a MAIL command may be given as a parameter.

Return from viewing the help text to MAIL using Q or QUIT.

```
Mail:HELP
•
•
View: QUIT

Mail:HELP COMPOSE
•
•
View: QUIT
```

Mail:LIST <messages>/<file or device>

This command displays messages in the current folder on the console, or alternatively lists to a file or device. In the latter case, a SCAN (see below) is prepended to the listing. If <messages> is omitted, the current message is listed.

```
Mail:LIST  - displays the current message on the console

Mail:LIST NEW,DRAFT/.LP     - lists all new messages plus the draft
                              to the line printer
```

If a message listed contains a "Mode:" component, this indicates that the body of the message contains a binary file or an EMAS file complete with header - hence the "Text:" component of the message is not listed. The "Text:" component may be reassembled as a file using the OUTPUT command.

Note that as a side effect of LIST, the status of a NEW message is changed to OLD.

Mail:NEXT

This command LISTs on the console the first undiscarded message after the current message. (Note the difference in meaning between this and the "next" message-reference keyword). The message listed becomes the current message.

Mail:OPEN <folder name>, NEW

This command switches primary attention to another folder, i.e. makes the folder named the current folder. It is also used to create a folder; in this case the second parameter NEW must be given. If no parameter is given, the standard folder M#INBOX is made the current folder.

```
Mail:OPEN BUGS,NEW - creates a new folder BUGS and makes it the
                     current folder

Mail:OPEN F2       - makes an existing folder F2 the current
                     folder

Mail:OPEN          - makes M#INBOX the current folder.
```

If the parameter '?' is given, then the name of the current folder and the number of messages in it is printed:

```
Mail:OPEN ?
Folder M#INBOX contains 16 messages
```

Mail:OUTPUT <component>/<file or device>

This command is used to transfer a single component of a message to a file or device or to the console. The default component is the "Text:" component and the default message the current message. If the "Text:" component of a message is to be output, then a check is made on whether a "Mode:" component is also present: for a "Mode:" value of BINARY, a store map file is output; for a "Mode:" value of EHEADER, an EMAS file complete with header is output.

Mail:OUTPUT	- displays the "Text:" component of the current message on the console.
Mail:OUTPUT CC:4(F3)	- displays the "cc:" component of message 4 in folder F3 on the console
Mail:OUTPUT /OBJECT1	- outputs the "Text:" component of the current message to a file

Mail:PREVIOUS

This command LISTs on the console the first undiscarded message prior to the current message. (Note the difference in meaning between this and the "previous" message-reference keyword). The message listed becomes the current message.

Mail:QUIT

Exits from MAIL and returns to Subsystem command level.

Mail:REPLY <message>

This command provides a convenient way of replying to a received message.

If no parameter is given then a reply to the current message is produced. A prompt is issued for the "Text:" of the reply. The escape character '@' can be used at this point to indicate input from an EMAS file or from an existing message component:

```
Mail:REPLY 2
Text:
:Your message received
::
Send or send-and-file-a-copy now?
Y or Fcc or N: N
```

```
Mail:REPLY LAST(FOLDER3)
Text:
:@ACK_VPOLITE
Send or send-and-file-a-copy now?
Y or Fcc or N: Y
```

If a <message> in the current folder is specified, it is made the current message.

Mail:RETRIEVE <messages>

The complement of DISCARD, this command changes the status of the specified messages in the current folder from "discarded" to "old". Note that once discarded the draft message cannot be retrieved. If <messages> is null, the current message is retrieved.

The first message in <messages> becomes the current message

```
Mail:RETRIEVE DISCARDED

Mail:RETRIEVE 4,10-12
```

Mail:SCAN <messages>/<device or file>

This command scans the specified messages in the current folder and produces a "list of contents" - a series of one line summaries for the messages. By default, NEW messages are scanned and the output is directed to the console.

The format of the one line summaries is as follows:

status	-	null	= old message
		s	= saved draft message (created when a draft message is FILEd)
		x	= discarded message
		*	= the draft message
		n	= new messages, i.e. messages not yet LISTED
index	-	the index number of the message within the folder	
<=	-	for the current message, the indicator '<=' is printed	
(length)	-	the length of the "Text:" component of the message in bytes	
date	-	the day and month that the message was sent	
from	-	the "From:" component of the message; if this component is empty, the "To:" component is displayed (prefixed with "To:")	
subject	-	the "Subj:" component of the message; if this component is empty the first few bytes of the "Text:" are displayed.	

Mail:SEND <message>, FCC/<folder name>

This command packages up a message and submits it for transmission. If <message> is omitted, the draft message is sent.

The FCC parameter indicates "File carbon copy", and causes a copy of the message to be filed in the current folder or the specified folder (this action is conditional on the message having been sent successfully). If the message sent lies within the current folder then it becomes the current message.

```
Mail:SEND , FCC          - sends the draft and files a copy in the
                           current folder

Mail:SEND 1(STANDARD) - sends the first message in folder STANDARD
```

Mail:STOP

Exits from the MAIL program and returns to Subsystem command level. STOP is identical to Mail:QUIT.

Mail:TIDY <folder name>

This command causes discarded messages to be purged from the indicated folder (by default, the current folder). TIDY is irreversible.

The remaining messages in the folder are ordered by transmission date (in the case of SAVED draft messages, by date of filing), and hence the message index numbers change.

If the current folder is tidied, the current message becomes the first in the folder.

```
Mail:TIDY F2
Mail:TIDY
```

EMAS 2900

New HELP facility

After lengthy discussion with advisors and others in the Centre I have decided that there is a continuing role for the HELP command. I see it as a very simple to use source of first level information on a wide variety of topics. I propose to change the user interface slightly in the following ways :

* reduction to a single parameter - a keyword - hence removal of mechanism to send output elsewhere.
* removal of facility to list all entries HELP (.ALL).
* generalisation of keyword structure - no longer restricted to valid membername. This allows e.g.

```
HELP COMMAND PARAMETERS
HELP .LP                (information about .LP)
```

* a facility may be included to attempt to provide possible matches for unknown keywords.

Apart from these changes in appearance the system will monitor all calls - particularly un-successful calls to determine what subjects users are asking about, for which we currently provide no information.

Improvement to Information

The changes listed above are minor compared to the changes to the information on which the system relies. It is proposed to expand this vastly to include references to many more topics. For many entries the information returned will be the source of further information - e.g. PACKHELP, MICROAID etc. Enough information must be given for the novice user to get to the further information.

The aim will be to provide brief, concise information and it is suggested that a maximum of 10 lines of 72 characters be provided per entry. This will be appropriate for use on slow terminals as well as faster terminals.

Development

Brian Murdoch will be responsible for providing the software for the routine HELP itself and for any tools required to maintain the data base. Information for entries will be collected from the existing HELP system, and a wide variety of other sources by Roderick McLeod and Nick Stroud. Advisors will be asked via ADLIB to suggest areas and to provide information on the specialities.

Conclusion

This development is thought to be a necessary step in the provision of a simple, single entry, source of information on a wide variety of EMAS related topics. It is consistent with packages such as MAIL, VIEW, SCREED which all provide their own internal HELP information, in that it will provide just enough information for the user to get to the facility concerned. In the case of the basic EMAS commands it will continue to provide a main source of information - backed up in the case of the more complex commands with references to the Users Guide.

Roderick McLeod

SCREED: A Screen Editor

11/2/81

Introduction

SCREED is a screen editor available on EMAS at ERCC. Details of access and a list of suitable video terminals are given in Appendix A.

The purpose of SCREED is to provide a simple means of editing character files. It does this by exploiting the fact that a video screen has 24 lines (usually) and that it is possible to move the "cursor" (current position marker) to any position on any of the lines. When characters are typed thereafter they appear on the screen at the cursor position.

Now, if a video screen were filled with lines of a character file, the effect of moving the cursor to some position on the screen and then typing would be to overwrite the text at that position. If a text editor running on EMAS had caused the file lines to appear on the screen in the first place, then this editor could note the user's subsequent movement of the cursor and when the user overwrote some of the text on the screen, it could make an exactly similar change to the original file.

The main advantage of this approach to editing over that used by, say, EDIT or ECCE is that instead of typing a "command string" to cause the editor to change the file, the user is "changing it himself", or so it appears, and the effect on the file is immediately obvious.

This summary of the principle of screen editing leaves a number of questions unanswered. For example:

* How do I tell the editor which part of my file is to be displayed on the screen?

* What happens if the text to be added at some point in the file is smaller (or larger) than the text to be removed?

* The cursor control keys on my terminal (the "arrow keys") do not work with EMAS.

To take the last point first: the communications network rejects most of the "control" characters if you type them on your keyboard. However, the network has been modified to allow a new mode of operation in which all characters are acceptable. (This is necessary because different types of video do the same thing, such as moving the cursor around, in quite different ways, and so it is not possible to identify a set of "cursor moving" control characters.) This new mode of operation is automatically selected within SCREED.

To delete text without replacing it by new text, a delete character is provided. This is '^' by default, but it can be changed. To add text to the file you need to use "control commands", described below.

Before reading further, you are advised to log on to EMAS and invoke SCREED (refer to Appendix A for details of access):

```
Command:SCREED(input file, output file)
```

The file specification is as for ECCE. Choose as input an unwanted file, or a copy of a file. SCREED will ask you to indicate which type of video terminal you are using. Once this has been replied to, the screen will go blank and then the first two lines of the input file will appear at the top of the screen. Holding the 'control' button down, type 'R' followed by 'S'. The first screenful of the input file will appear on the screen. You will find that the cursor moving buttons will operate - line feed, return, up arrow, down arrow, backspace, home, etc.

TO TERMINATE AN EDIT SESSION, TYPE
STOP followed immediately by ctrl X (ctrl Z on some terminals)
or ABORT followed immediately by ctrl X (ctrl Z on some terminals)
(Details of this type of edit command are given below.)

[Note that '(A)' in the description below stands for control A, i.e. the code generated by holding down the control button and typing 'A'. Similarly (B), (C), etc.]

When SCREED is being used, there are no "command line" or "menu" areas on the screen, and - apart from delete characters - no characters stand for other than themselves: all that you see is file. However, some blank lines on the screen may not correspond to blank lines in the file. Such screen lines are called "null".

To move the cursor to a specific part of the screen the user can make use of whatever keys are available on his keyboard: Tab, Backspace, Return, Line Feed, |, |, <-, ->, Home - but not the space bar since it inserts a space character at the current position. In addition to those keys there are cursor positioning commands provided by SCREED, as described below.

The user commands SCREED by means of pairs of control characters. The first indicates an action and the second the text unit to which the action applies. If the "action" control is omitted the last action control given is assumed.

Action		Text Unit
(A)dd (B)eginning of (C)onjoin (D)isjoin (E)nd of (O)mit (N)ext (P)revious (R)ewrite (T)runcate	followed by	(W)ord (L)ine (S)creen

Any combination of an action followed by a text unit may be given, but may not be valid in some circumstances. For example, (O)(W) (Omit Word) will fail if the cursor is not on a "word" (a piece of text delimited by spaces or the beginning of a line or the end of a line). The editor signals failure by wiggling the cursor back and forth on the current line.

It is probably simpler to try out the effects of these commands on a file than to read a detailed description of their effects. Explanatory notes on some of the combinations may be useful, however:

1) Some commands move the cursor only, some delete text and some rewrite the screen completely.

2) (O)(L) turns the current line into a null line.

3) (O)(W) causes the current word and appropriate space characters beside it to be replaced by delete characters.

4) (R)(L) removes all delete characters from the current screen line in the process of rewriting it.

5) (D)(L) causes the current line to be broken ("disjoined") at the cursor. In general the remainder of the screen has to be rewritten to accommodate the new line produced.

6) (A)(W) shifts the remainder of the current line about 15 places to the right and fills the gap with delete characters. These can then be overtyped as required. If the end of the line was moved off screen in the process, it reappears on (R)(L), although you do not have to follow (A)(W) with (R)(L). (Indeed, there is no command which must be followed by some other specific command - they are all entirely independent.)

7) (A)(L) causes the current line and all the lines below it to be replaced on the screen by null lines, the current line then to be rewritten at the bottom of the screen and finally the cursor to be moved back to the first of the null lines. These null lines can then be used as necessary. Null lines below the last line added will be discarded on typing (R)(S), but note that null lines traversed prior to text being added lower down will be turned into blank lines (i.e. they become part of the file).
If your terminal is operated by SCREED in page mode (e.g. the Lynwood DAD-1), then (A)(L) causes the screen to be blanked, the line previously containing the cursor to be written at the bottom of the screen and the line before that to be written at the top of the screen. The screen lines between are all null.

8) (A)(S) is similar to (A)(L) except that the current line is not rewritten at the bottom of the screen. Thus an arbitrarily large number of lines can be added simply by scrolling the screen when you get to the bottom. Use of a "clear screen" key (if your terminal has one) is equivalent to (A)(S) - the resulting screenful of null lines is understood to come before the line which previously contained the cursor.
If your terminal is operated by SCREED in page mode, (A)(S) causes the screen to go blank, as though you had used a "clear screen" key (as described above).

9) (R)(S) Rewrites the screen, removing all delete characters and null lines. If there are null lines above the cursor when (R)(S) is typed then file lines prior to those currently displayed are brought down onto the screen; if there are null lines below the cursor when (R)(S) is typed then file lines following those currently displayed are brought up onto the screen. The only exception to this rule is when there are insufficient previous file lines to bring down: in this case the first line of the file appears at the top of the screen. On

the other hand, if there are insufficient file lines following those displayed to bring up onto the screen, then null lines appear at the bottom of the screen.

10) (D)(S) means that the screen is to be "disjoined" at the current line. The current line becomes the first of the new screen; i.e. the screen scrolls until the current line is at the top. In the case of page mode terminals, the screen is rewritten with the current line at the top.

11) (C)(S) implies that this screenful and the next are to be "conjoined". The resulting screenful shows the join exactly in the middle of the screen; i.e. the effect is to scroll up by half a screen. Again, to achieve the required effect on page mode terminals, the screen is rewritten to display the appropriate text.

12) (O)(S) causes all the lines displayed on the screen to be removed ("omitted") from the file. (T)(S) is similar: it removes the current line and all lines below it on the screen from the file.

13) (N)(S) causes the file line following the bottom one currently displayed on the screen to become the top line on the screen.

Having read this, you should now experiment with the various combinations of action controls and text units. A summary of commands appropriate to your video type is given in Appendix B.

These facilities are useful, but they are not sufficient. The following questions make this clear:

* How do I terminate (or abandon) the edit session?

* How do I get to the middle of a large file (typing (N)(S) fifty times is rather tedious)?

* Can I save the editing done to date?

* How can I correct twenty occurrences of the same mis-spelled word?

* Can I change the delete character?

All these facilities are provided by means of another control character, (X) [(Z) on some terminals - see Appendix B]. If you type in some text and follow it with (X), SCREED rewrites the line as it was before you typed the text, and then examines the text typed. If it is one of the following, in upper or lower case, then a special action is implied:

CLOSE (or STOP or END or EXIT)
ABORT
SAVE
EDIT
ECCE
DEL=<char>

If the text was none of these then SCREED takes no action.

CLOSE
STOP
END
EXIT

Any of these causes the edit session to be terminated.

ABORT — The edit session is terminated without the output file being written to.

SAVE — The file is copied in its present state to the output file, the screen is blanked and then the current line and the line before and line after are written in the middle of the screen (the other screen lines are null).

EDIT — A branch is made to the Subsystem editor EDIT. The current position is preserved. You can terminate the edit session from within EDIT in the usual way, or return to SCREED if you wish. This is done by hitting the ESC key and replying "SCREED" (or "screed") to the "INT:" prompt. On hitting the "Return" key the "Edit:" prompt will appear as usual, but if you hit "Return" again SCREED will be reverted to. Three lines will be displayed, as for SAVE.

ECCE — A branch to the editor ECCE is made. The return to SCREED, if desired, is precisely as for the return from EDIT.

Note that you can start your editing in ECCE or EDIT and then branch to SCREED by the use of the "INT:". At present you must invoke EDIT from command level by the command TEDIT, and ECCE by TECCE.

DEL=<char> — The specified character becomes the delete character in future.

Thumbing

To move around the file you could invoke EDIT or ECCE by use of (X), then move the current position, then return to SCREED. However you can also do this without leaving SCREED, as follows: suppose that the top line of the screen corresponds to the first line of the file and that the bottom line of the screen corresponds to the last line of the file; then halfway down the screen ≡ halfway down the file, three quarters down the screen ≡ three quarters down the file, etc. If you position the cursor on the left hand margin of a screen line and then type (X), the effect is to move to the corresponding position of the file. The screen is blanked and three lines from the implied file position are written, at the specified screen position. If this is not in fact the right place then you can adjust your position, up or down, and type (X) again. This is known as "thumbing", as you are effectively thumbing through the file.

The reason for SCREED writing only two or three file lines, on entry or following a (X) command, is to save time - you might want to "thumb" to some other part of the file, and writing a complete screenful of text would thus waste time. You can always use (R)(S) if you want to see a screenful; in this case, note the significance of the cursor position, as explained in note (9) above.

Other notes

* The Recall file is not written to during the operation of SCREED.

* If the terminal seems to go dead and not even (X) has any effect, there is an "INT:" character which you can resort to: (]) with most terminals - see Appendix B. Note, however, that using it causes you to lose all your editing.

* All constructive comments and suspected faults should be directed to me.

John Murison
ERCC
(667 1081 ext. 2639)

On 2972 or 2980:

```
Command:INSERT(KNTLIB.SCREEDY)
Command:INSERT(KNTLIB.ECCE24)
Command:INSERT(KNTLIB.EDIT27)
```

The command to invoke SCREED is:

```
Command:SCREED(input file, output file)
```

Either parameter (but not both) may be omitted.

The use of the parameters is as for ECCE. The only difference between the SCREED and EDIT parameters is when specifying that a new file is to be edited:

```
Command:EDIT(FRED)
Command:SCREED(,FRED)
```

are equivalent in this case.

Terminals supported

SCREED indicates which terminals it can support when you call it. At the time of writing these are:

Perkin Elmer 550

Newbury 7000 Series

Lear Siegler ADM-3A

Lynwood DAD-1

Visual 200

Pericom 6801

Relevant details of each of these terminals are given in Appendix B.

Note that some older video terminals are not suitable for use with SCREED, for example the ITT 3210.

Please contact me if you have a terminal which is not in the above list and which you think might be suitable for use with SCREED.

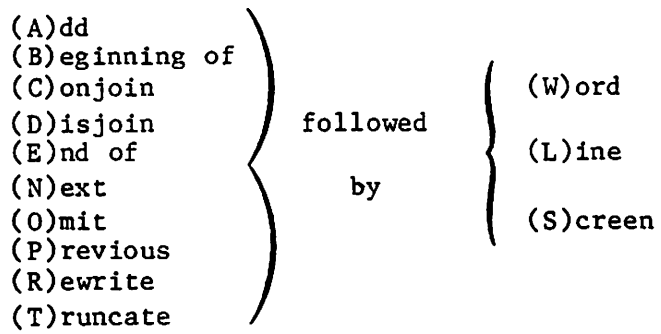
> WARNING
>
> This is a preliminary version of SCREED. It may be changed without warning. It may contain errors which corrupt files.

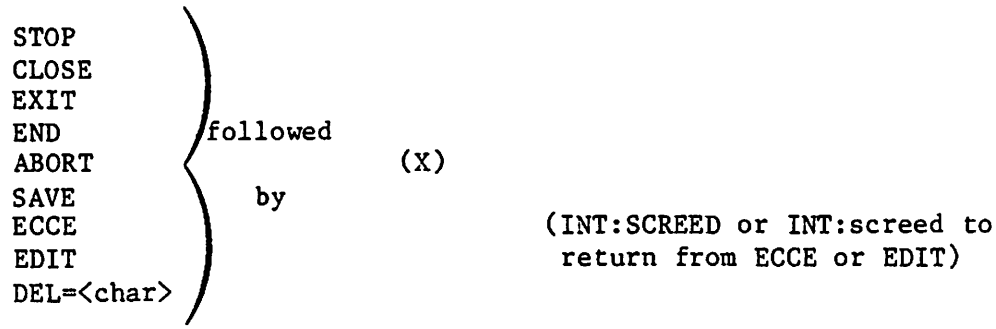
APPENDIX B-1: Perkin-Elmer 550

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctrl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.





You can "thumb" through the file by use of (X) on the left hand margin.

(]) in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the Perkin-Elmer 550

* There are no "arrow" keys. Thus the cursor has to be moved by the use of Tab, Backspace, Return, Line feed, and the appropriate SCREED commands (the (W) commands in particular).

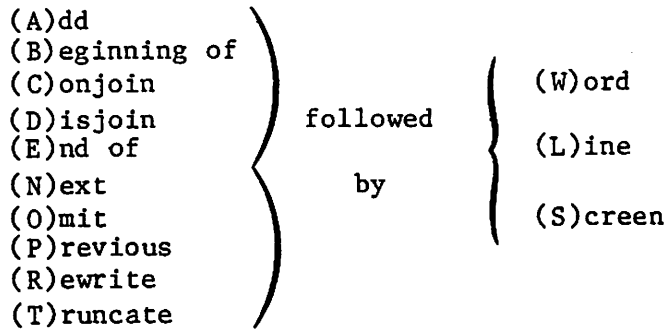
* The Clear key (which only operates when Ctrl is depressed also) does not transmit any code - the effect is entirely local to the terminal. Thus if you clear the screen by using it, SCREED is not aware of this. Therefore it should not be used.

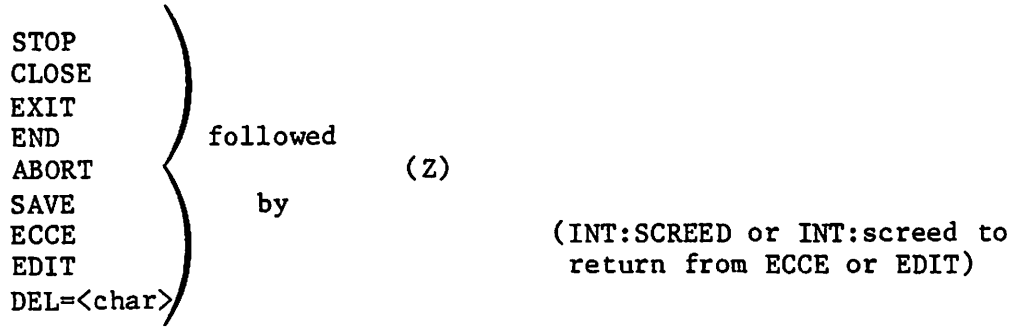
APPENDIX B-2: Newbury 7000 Series

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctrl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.





You can "thumb" through the file by use of (Z) on the left hand margin.

ESC in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the use of Newbury 7000 Series terminals

* As stated above, (Z) is used rather than (X), and ESC is the "INT:" character rather than (]).

* (With respect to the 7001:) The keyboard has a separate row of keys along the top. Most of these are either disabled or do not generate any codes. However the keys from Home rightwards are usable.

* If the terminal does not seem to be operating correctly when used with SCREED, it might not be set up appropriately. This can checked, and corrected if necessary, as described below. Note that during the procedure the terminal is effectively off-line, and so does not interfere with any on-line work in progress.

  Press the Edit key (which is not normally used), and then Ctrl and Home together. A screenful of text appears; it details various settings (which on most other terminals are determined by switches on the terminal). When you press Tab the cursor moves from one setting to the

next. If you press Return repeatedly the various possible values of the current option appear in turn.

The settings relevant to the operation of SCREED are as detailed below; the other settings should not be changed:

First line:

```
    ..............BELL=80;DELETE=N;NL=NO....     (BELL=80 turns off bell)

                    TELETYPE = NO
    :
    :
```

Under the heading ATTRIBUTES:

```
    PROTECT = NO
    BLINK = NO
    UNDERLINE = NO
    :
    :
```

Of the codes given under the heading FROM LINE, the following must be underlined:

            0C 1A

and the following must not be underlined:

            08 09 0A 0B 0D 16 18 19 1D 1F

The settings of the other FROM LINE codes is immaterial to the operation of SCREED.

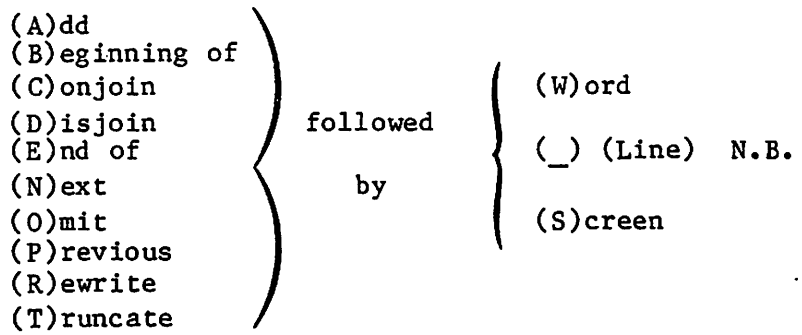
You should then finish by pressing ESC followed by Edit. This puts you back on-line to EMAS.

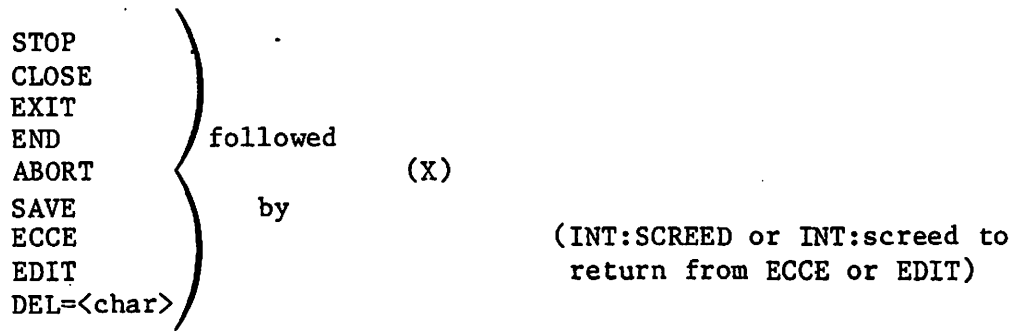
APPENDIX B-3: Lear Siegler ADM-3A

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.





You can "thumb" through the file by use of (X) on the left hand margin.

(]) in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the Lear Siegler ADM-3A

* As indicated above, (_) is used instead of (L).
* Although the terminal has a line width of 80 characters, SCREED (for good reasons) does not use the 80th column of any line, and will ignore any character typed in the 80th column.
* There is a (standard) option on the ADM-3A whereby typing the space bar does not always generate a space character - it merely moves the cursor right. This is not appropriate to the operation of SCREED and should therefore be changed. The person responsible for the terminal should be consulted, as the option switch in question is under the terminal casing. The instructions for making the change are described on page 2-3 of the terminal handbook. The switch in question is located at the top of the left-hand set of toggle switches; it is labelled "SPACE 6 / ADV"; it should be set to SPACE (by default it is set to ADV).

APPENDIX B-4: Lynwood DAD-1

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.

N.B.

(A)dd
(B)eginning of
(C)onjoin
(D)isjoin
(E)nd of
(F)ollowing
(O)mit
(P)revious
(R)ewrite
(T)runcate

followed by

(W)ord
(Q) (Line)
(S)creen

N.B.

STOP
CLOSE
EXIT
END
ABORT
SAVE
ECCE
EDIT
DEL=<char>

followed by (X)

(INT:SCREED or INT:screed to return from ECCE or EDIT)

You can "thumb" through the file by use of (X) on the left hand margin. (]) in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the Lynwood DAD-1 terminal

* Note that, as stated above, (F) (Following) is used instead of (N) (Next), and that (Q) is used instead of (L) (Line).

* SCREED operates the Lynwood in page mode, which it selects automatically. This affects the operation of (C)(S), (D)(S) and (A)(L), as explained in the description of SCREED.

* The Line Feed key has been disabled, as it has the effect of clearing any line that it causes the cursor to move onto.

* The two switches at the back of the terminal marked St.CR and St.LF should both be down (i.e. off).

* Sometimes on entry to SCREED, the first two lines of the file appear at the bottom of the screen rather than the top. This has to do with the terminal warming up (I think). SCREED will operate as though the two lines - and the cursor - were at the top of the screen. It is therefore best in this situation either to "thumb" or to use (R)(S) or (F)(S).

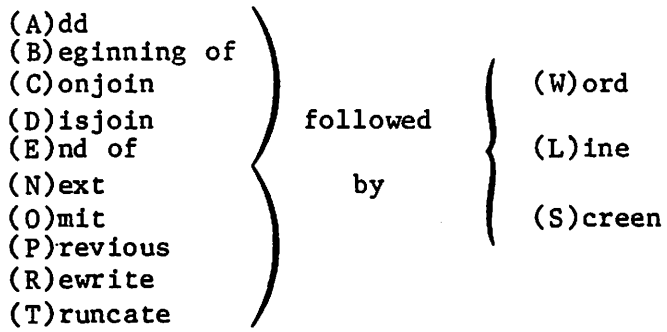
* Although the terminal has a line width of 80 characters, SCREED (for good reasons) does not use the 80th column of any line, and will ignore any character typed in the 80th column.

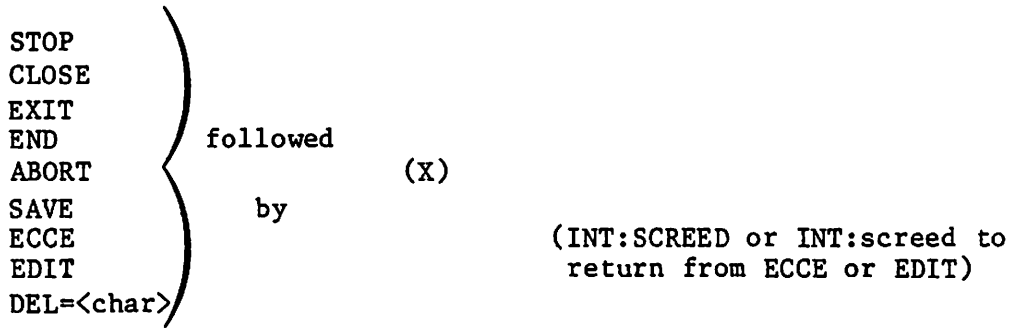
APPENDIX B-5: Visual 200

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.





You can "thumb" through the file by use of (X) on the left hand margin.

(]) in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the Visual 200 terminal

* The Visual 200 terminal has a large number of features and settings; each is activated by the user generating - either explicitly or by use of a function key - a string of characters starting with ESC. SCREED can detect when this has been done but unless the feature happens to be one that SCREED uses (such as "Clear screen" or "Clear rest of line"), it will generate an error message and rewrite the screen. This however might not be enough, and to ensure the proper working of SCREED thereafter the user might have to reset the terminal; e.g. if he had issued the command to "clear all tab settings" SCREED would not be aware of this and would operate as though the default tab settings still applied.

* The "typo-matic" keys of this terminal make it particularly suitable for use with a screen editor.

APPENDIX B-6: Pericom 6801

Command summary

Delete character is '^'. It can be changed (see "DEL=" below).

(A) below stands for Ctl A (i.e. hold down the control button and type 'A'). Similarly (B), (C), etc.

(A)dd
(B)eginning of
(C)onjoin
(D)isjoin
(E)nd of
(N)ext
(O)mit
(P)revious
(R)ewrite
(T)runcate

followed by

(W)ord
(L)ine
(S)creen

STOP
CLOSE
EXIT
END
ABORT
SAVE
ECCE
EDIT
DEL=<char>

followed by (X)

(INT:SCREED or INT:screed to return from ECCE or EDIT)

You can "thumb" through the file by use of (X) on the left hand margin.

(]) in SCREED is the "INT:" character. All editing is lost if you use it.

Notes relating to the Pericom 6801 terminal

* SCREED requires the terminal to be in page mode. The user must set the page switch, beside the screen, at some stage before replying to SCREED's initial question about the terminal type.

* The Auto CR/LF and Auto LF wheels at the back of the terminal should both be rotated upwards to the limit, i.e. switched off.

* The terminal has arrow keys, on a separate pad. Unfortunately, although they move the cursor around the screen, they do not generate any codes and so SCREED is unaware that the cursor has been used. In other words, DO NOT USE THE ARROW KEYS.
(The cursor has to be moved by the use of Tab, Backspace, Return, Line Feed and the appropriate SCREED commands, the (W) commands in particular.)

* Although the terminal has a line width of 80 characters, SCREED (for good reasons) does not use the 80th column of any line, and will ignore any character typed in the 80th column.

DOC/EMAS.K3.0/1

University of Kent at Canterbury	COMPUTING LABORATORY
Compiler PARM settings and their meanings R. D. Eager	
Date April 1980	Latest Revision (1) January 1984

Contents

1 Introduction

2 ALGOL options

3 BCPL options

4 FORTRAN (G) options

5 FORTRAN 77 options

6 IMP80 options

7 Miscellaneous options

8 Acknowledgements

University of Kent at Canterbury © 1980

1 Introduction

The Subsystem command "PARM" is used to set compiler options for subsequent compilations; see the "EMAS 2900 User's Guide" for further details.

This document describes how various compilers use the PARMs, and which are selected by default. The various options come in pairs; at any time, only one of each pair may be in force. Generally, only the first option in each pair may be given as a parameter to the PARM command; the other is selected whenever the first option is not quoted. The exceptions to this are the QUOTES and R8 options; these are the default, but are also the keywords recognised for that option. Users requiring R4 or PERCENT options should consult the author of this document.

In the following Sections, the normal default setting is thus given second.

2 ALGOL options

Option pairs		Meaning
NOLINE	LINE	LINE causes code to be included to keep a record of the number of the line being executed, for diagnostics.
EBCDIC	ISO	specifies the code to be used for internal character data.
DYNAMIC	STATIC	DYNAMIC marks all external references for satisfying dynamically.
FIXED	FREE	FIXED causes all characters past column 72 in the program source to be ignored. FREE gives no limitation.
OPT	NOOPT	OPT gives rather less checking than NOCHECK, NOLINE, NODIAG (e.g. stack overflow is not checked).
CODE	NOCODE	CODE causes a listing of the object code generated by the compiler to be produced.
NOTRACE	TRACE	TRACE causes routine traceback information to be kept for diagnostics.
NOARRAY	ARRAY	ARRAY causes an array bound check to be carried out on all array accesses. Note that some checking on multidimensional array bounds will be carried out even if NOARRAY is selected.
NOCHECK	CHECK	CHECK causes a check to be made that each variable, whose value is about to be used, has been assigned a value.

NODIAG	DIAG	DIAG causes symbol tables to be retained at run-time, for diagnostics.
NOLIST	LIST	LIST causes a program listing to be generated at compile-time. NOLIST causes output of a heading and any error messages only.
PERCENT	QUOTES	specifies keyword notation. QUOTES means that keywords are in single quotes; PERCENT means that keywords are preceded by a percent sign.
MAXDICT	NORMALDICT	MAXDICT causes the compiler tables to be set to the maximum possible size.

3 BCPL options

Option pairs		Meaning
PARMX	NOPARMX	PARMX causes strings to be made read-only.
DYNAMIC	STATIC	DYNAMIC marks all external references for satisfying dynamically.
OPT	NOOPT	OPT gives NOTRACE, NOPROFILE.
CODE	NOCODE	CODE causes a listing of the object code generated by the compiler to be produced.
PROFILE	NOPROFILE	PROFILE causes extra code to be inserted to keep a count of how often each piece of code is executed. Call of routine MAPSTORE (automatic at end of run) then gives summary of information, and execution histogram.
NOTRACE	TRACE	TRACE causes routine names and traceback information to be kept, for diagnostics.
STACK	NOSTACK	STACK causes the APTOVEC function to take space from the hardware stack.
NOLIST	LIST	LIST causes a program listing to be generated at compile-time. NOLIST causes output of a heading, statistics and any error messages only.
XREF	NOXREF	XREF causes a program cross-reference listing to be generated at compile-time.

4 FORTRAN (G) options

Note that these options apply only to the FORTRAN (G) compiler for FORTRAN 66 programs; see Section 5 for details of the options for the FORTRAN 77 compiler.

Option pairs		Meaning
MISMATCH	NOMISMATCH	MISMATCH allows the size of a subprogram argument to be determined by the size of the corresponding dummy argument. NOMISMATCH forces the sizes of corresponding dummy and actual arguments to be the same (NOMISMATCH results in faster execution).
EBCDIC	ISO	specifies the code to be used for internal character data, and Hollerith constants.
DYNAMIC	STATIC	DYNAMIC marks all external references for satisfying dynamically.
OPT	NOOPT	OPT gives NOCHECK, NODIAG.
ATTR	NOATTR	At the end of each program unit, ATTR causes a list of attributes (type, size, whether scalar or array etc.) to be output.
CODE	NOCODE	CODE causes a listing of the object code generated by the compiler to be produced.
LABELS	NOLABELS	LABELS causes code to be included to maintain a list of the most recent labels, subprogram calls and returns executed, for diagnostic purposes.
XREF	NOXREF	XREF causes output of references to variables, with line numbers of their occurrences.
ZERO	NOZERO	ZERO causes COMMON areas, not initialised by a BLOCK DATA subprogram, to be filled with zeros. NOZERO causes them to be filled with the "unassigned" pattern.
INHIBIOF	ALLOWIOF	INHIBIOF inhibits the occurrence of an integer overflow interrupt. This facility may be withdrawn later.
NOARRAY	ARRAY	ARRAY causes a hardware check that all array subscripts lie within bounds at run-time. With NOCHECK (see below) selected, ARRAY checks upper bounds, but not all lower bounds. NOARRAY causes hardware array bound checking to be omitted.
NOCHECK	CHECK	CHECK causes a check to be made that all variables whose values are used, have been assigned a value. It also causes a check that the number of parameters passed to a subprogram is correct, and that array element references are in the total bounds of the array. NOCHECK omits all this checking, but also causes all variables established by the compiler (i.e. excluding COMMON blocks other than those initialised by a BLOCK DATA subprogram) to be initialised to zero.

NODIAG	NODIAG	DIAG causes symbol tables to be retained at run time, for diagnostics.
NOLIST	LIST	LIST causes a program listing to be generated at compile time. NOLIST causes only erroneous statements to be listed.
MINSTACK	NORMALSTACK	MINSTACK causes array dope vectors to be stored off the stack, thus reducing the amount of initialised stack required by the program.
MAXDICT	NORMALDICT	MAXDICT causes the compiler tables to be set to the maximum possible size.
R4	R8	Specifies the number of bytes to be allocated to real variables.
L8	L4	Specifies the number of bytes to be allocated to LOGICAL variables.
I8	I4	Specifies the number of bytes to be allocated to INTEGER variables.

5 FORTRAN 77 options

Note that these options apply only to the FORTRAN 77 compiler; see Section 4 for details of the options for the FORTRAN (G) compiler.

Option pairs		Meaning
EBCDIC	ISO	specifies the code to be used for internal character data, and Hollerith constants.
DYNAMIC	STATIC	DYNAMIC marks all external references for satisfying dynamically.
OPT	NOOPT	OPT inhibits all array bound checks and implies NOCHECK.
OPT1	NOOPT1	Implies OPT, and performs partial source level optimisation.
OPT2	NOOPT2	Full optimisation is to be performed.
OPTEXT	NOOPTEXT	If OPT1 or OPT2 has been specified, OPTEXT will produce a pseudo-FORTRAN listing of the optimised program.
NOCOMMENTS	COMMENTS	NOCOMMENTS suppresses the output of comment messages.
NOWARNINGS	WARNINGS	WARNINGS implies that warning messages are generated at compile-time when any features not in the ANSI FORTRAN 77 standard are used. NOWARNINGS suppresses the output of such messages.

STRICT	NOSTRICT	If STRICT is specified then an error is reported when any feature not in the ANSI FORTRAN 77 standard is used.
ATTR	NOATTR	At the end of each program unit, ATTR causes a list of attributes (type, size, whether scalar or array etc.) to be output.
CODE	NOCODE	CODE causes a listing of the object code generated by the compiler to be produced.
XREF	NOXREF	XREF causes output of references to variables, with line numbers of their occurrences.
INHIBIOF	ALLOWIOF	INHIBIOF inhibits the occurrence of an integer overflow interrupt. This facility may be withdrawn later.
NOARRAY	ARRAY	By default, FORTRAN 77 checks that each subscript in an array reference is within the declared range for that dimension. NOARRAY reduces the check to ensuring only that the reference is within the declared size of the total array.
NOCHECK	CHECK	By default, FORTRAN 77 checks that variables are assigned a value before being referenced and that actual and formal arguments are of corresponding size and type. NOCHECK inhibits these checks and implies NOARRAY.
NODIAG	NODIAG	DIAG causes symbol tables to be retained at run time, for diagnostics.
NOLIST	LIST	LIST causes a program listing to be generated at compile time. NOLIST causes only erroneous statements to be listed.
R4	R8	Specifies the number of bytes to be allocated to real variables.
L8	L4	Specifies the number of bytes to be allocated to LOGICAL variables.
I8	I4	Specifies the number of bytes to be allocated to INTEGER variables.
PROFILE	NOPROFILE	Set before compiling a program and/or any separate modules for which a profile analysis is required. The statement CALL FORTRANPROFILE(N) must have been placed before the last normally executed.

6 IMP80 options

Option pairs		Meaning
PARMY	NOPARMY	PARMY causes unused %constantintegers to be flagged as warnings. Since they generate no code, they are not normally flagged.
PARMZ	NOPARMZ	Used for debugging the compiler; causes the compiler to dump its workfile during compilation.
NOLINE	LINE	LINE causes code to be included to keep a record of the number of the line being executed, for diagnostics.
DYNAMIC	STATIC	DYNAMIC marks all external references for satisfying dynamically. Individual references are specified as dynamic by declaring them as dynamic.....spec rather than external.....spec.
FIXED	FREE	FIXED causes all characters past column 72 in the program source to be ignored. FREE gives no limitation.
DEBUG	NODEBUG	DEBUG causes code to be included to call the IMP interactive debugging facilities.
OPT	NOOPT	OPT gives rather more than NOCHECK, NOLINE, NODIAG (e.g. stack overflow is not checked).
CODE	NOCODE	CODE causes a listing of the object code generated by the computer to be produced.
PROFILE	NOPROFILE	PROFILE causes extra work to be inserted to keep a count of how often each line is executed. A call of the intrinsic routine PPROFILE then gives a summary of this information and resets the counts. PARM(LINE) is forced on.
NOTRACE	TRACE	TRACE causes routine traceback information to be kept, for diagnostics.
NOARRAY	ARRAY	ARRAY causes an array bound check to be carried out on all array accesses. Note that some checking on multidimensional array bounds will be carried out even if NOARRAY is selected.
NOCHECK	CHECK	CHECK causes a check to be made that each variable, whose value is about to be used, has been assigned a value.
STACK	NOSTACK	STACK causes arrays to be allocated on the hardware stack; recommended only for the experienced user.

NODIAG	DIAG	DIAG causes symbol tables to be retained at run-time, for diagnostics.
NOLIST	LIST	LIST causes a program listing to be generated at compile-time. NOLIST causes output of a heading and any error messages only.
MAXDICT	NORMALDICT	MAXDICT causes the compiler tables to be set to the maximum possible size.

7 Miscellaneous options

The following PARM option is effective other than at compile-time.

Option pairs		Meaning
MAP	NOMAP	MAP causes the LINK command to output a map of the linked file.

8 Acknowledgements

Much of the information in this document is derived from EMAS 2900 Subsystem Note No.7, dated 23.8.79. The authors of that Note were G. E. Millard, J. M. Murison, A. Shaw and P. D. Stephens, all of Edinburgh Regional Computing Centre.

Supervisor

EMAS 2900 SUPERVISOR NOTE

No: 1 (revised)
Date: 16/4/81

PON Mechanism for Director

Each process has four service numbers associated with it:

process+N1, process+N2, process+N3, process+N4

Currently, N1=64 (X'40'), and N2, N3 and N4 vary with the configuration. Service 'process+N1' refers to the Local Controller and should not be used by anyone but P.D.S.!!! N2, N3 and N4 may be found from COM_SYNC1DEST, COM_SYNC2DEST and COM_ASYNCDEST respectively (see Supervisor Note 15).

Services 'process+N2' and 'process+N3' refer to the Director. They are identical in the way they are handled by the Kernel and the Local Controller but the conventional use is intended to be that 'process+N2' should be the normal service for passing messages to the Director and that 'process+N3' should be used when a specific message (usually a reply) is being awaited. 'process+N4' also refers to the Director but is unlike the other two in that messages received on this service will be dealt with immediately, rather than waiting for a correspnding Director POFF. The normal effect is to enter Director's SIGNAL state.

OUTs 5 and 6 refer to service 'process+N2', OUTs 7 and 8 refer to service 'process+N3', and OUTs 9 and 10 refer to service 'process+N4'. OUTs 5, 7 and 9 imply that the process is to be suspended until a suitable message is available, and OUTs 6, 8 and 10 imply that the process wishes to continue processing after the PON.

All the OUTs expect the service to be PONned to be in DIROUTP. If DIROUTP_DEST>>16 is zero then no PON is done. This gives a pure 'suspend' for OUTs 5, 7 and 9. For OUTs 6, 8 and 10 with a DEST of zero it gives a TOFF. That is, a POFF(DIROUTP) is done on the appropriate service number. If no parameters are waiting, DIROUTP_DEST is set to zero. The source service number is inserted by the Local Controller, leaving the source activity unaltered. The inhibit/uninhibit mechanism operates so that 'process+N2' for OUT 5, and 'process+N3' for OUT 7, are uninhibited only when the process is suspended (the alternate service remains inhibited). The 'process+N4' service number is always uninihibited whenever the process is suspended.

Summary of effects of these OUTs

	Message type	Effect	
		DEST non-zero	DEST zero
OUT 5	SYNC1DEST	PON+POFF	POFF
OUT 6		PON (and continue)	TOFF
OUT 7	SYNC2DEST	PON+POFF	POFF
OUT 8		PON (and continue)	TOFF
OUT 9	ASYNCDEST	PON	-
OUT10		PON (and continue)	TOFF

It is also possible, through the use of OUTs 5-10, to send a message asynchronously to a process identified by means of its user name. This is known as the 'relay' facility.

To use it, DIROUTP_DEST should be set to X'FFFF' and the user name of the destination process should be loaded, left justified in ACC (128 bits, i.e. ACS=3), as a string(6). The Local Controller searches for this user name and replaces the X'FFFF' with the appropriate service number before doing the PON. The destination and source activities are unchanged, as standard. The source service numbers are also set as standard.

If the user name cannot be found, i.e. nothing logged on under that name, DIROUTP_DEST is set to zero and control is returned immediately to the Director. Otherwise, DIROUTP_DEST will be non-zero on return.

The relay facility allows messages to be PONned to any of the three local process service numbers. An additional parameter is required which takes the values 1, 2 or 3 to specify the local service number, the alternative local service number or the asynchronous service number, respectively. These are the analogue of the reply service numbers generated by OUT 5 and OUT 6, OUT 7 and OUT 8 or OUT 9 and OUT 10 respectively. This parameter is expected in the rightmost word of the accumulator. For example:

6	E	C	S	C	0	4	inc no		3

To enable different incarnations of a process to be distinguished, the incarnation number, in binary, can be appended to the string (as shown).

PON and Wait

An OUT 11 has been introduced which allows a message to be PONned and the process to remain in store until a reply arrives. Needless to say, in view of the possible blockage, this will need to be used very circumspectly; in particular for things like magnetic tape transfer, which one expects to take place quickly.

Due to the way this has to be implemented (using the Local Controller service number as the reply destination) there is no possibility of specifying any of the alternative local service numbers as a reply service number. The implications of this are clear, I hope. Also, because of the possibility of blockage and deadly embrace, only Kernel services can be PONned in this way and not messages to processes. The 'relay' facility is therefore not relevant here.

Delayed relay service number 63

Any standard parameter record ponned to service 63 activity N will be relayed, after a delay of N seconds, to the DEST/DACT in P_P6. P_SRCE, P_P1, P_P2, P_P3, P_P4 and P_P5 are left unchanged. N=0 is acceptable, if unwise.

These facilities are available from SUP26H onwards.

D.J. Rees
P.D. Stephens
J.K. Yarwood

EMAS 2900 SUPERVISOR NOTE

No: 2 (revised)
Date: 21/4/81

Data Passed to Director on Initial Entry

When Director is first activated, the B register contains the address of the Director / Local Controller parameter-passing record DIROUTP. This register is preserved by the Dirloader and passed onto Director as a recordname parameter.

DIROUTP contains useful information, including the address of the signal record SIGOUTP, which has further initialisation data.

Both records have the standard format:

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

They are initialised as follows:

DIROUTP

_DEST = Supervisor identifier, as string(3), e.g. "25F"

_SRCE = epage size <<16 ! max disc block size

_P1 = process number

_P2 and _P3 = username, as string(6), e.g. "MANAGR", followed by a byte giving the incarnation number.

_P4 = address of SIGOUTP

_P5 = address of System Call Index Table

_P6 = DACT to be used for terminal INT messages

SIGOUTP

_DEST = LST LEN, i.e. number of segments in Local Segment Table

_SRCE = address of Secondary Segment Table

_P1 = index of last Claimed Block Table entry

_P2 = address of Claimed Block Table

_P3 = address of accounting record

_P4 = address of instruction counter revolutions count

_P5 = address of Input/Output status record

_P6 = not used

P.D. Stephens

EMAS 2900 SUPERVISOR NOTE

No: 3 (revised)
Date: 21/4/81

Tables and Formats Describing the VM

The VM is described by a series of tables indexed from the Secondary Segment Table, which parallels the hardware Segment Table.

The formats of the tables are as follows:

SST byteinteger (0:segment table length-1)

CBT (integer DA, halfinteger AMTX, byteinteger TAGS, TAGS2)

DA		disc address of (start of) block
AMTX		index to Active Memory Table for block (valid only when active bit set in TAGS)
TAGS	bit 7	new, i.e. all pages in this block are initially zero
	bit 6	advise disc only (i.e. drum not to be used)
	bit 5	active, i.e. AMT entry allocated
	bits 4->0	limit (in epages) of block. Range 0-31, i.e. blocks up to 128 Kbytes
TAGS2	bit 7	continuation, i.e. this is not the first block in the file
	bit 6	advisory forward sequential - Supervisor may page the block assuming (repeated) forward sequential access
	bits 5->0	unused

AMTHASH ownhalfintegerarray (0:511)

list heads of chains of AMT entries whose disc addresses hash to the same value

AMT (integer DA, halfinteger DDP, LINK, USERS, byteinteger LEN, OUTS)

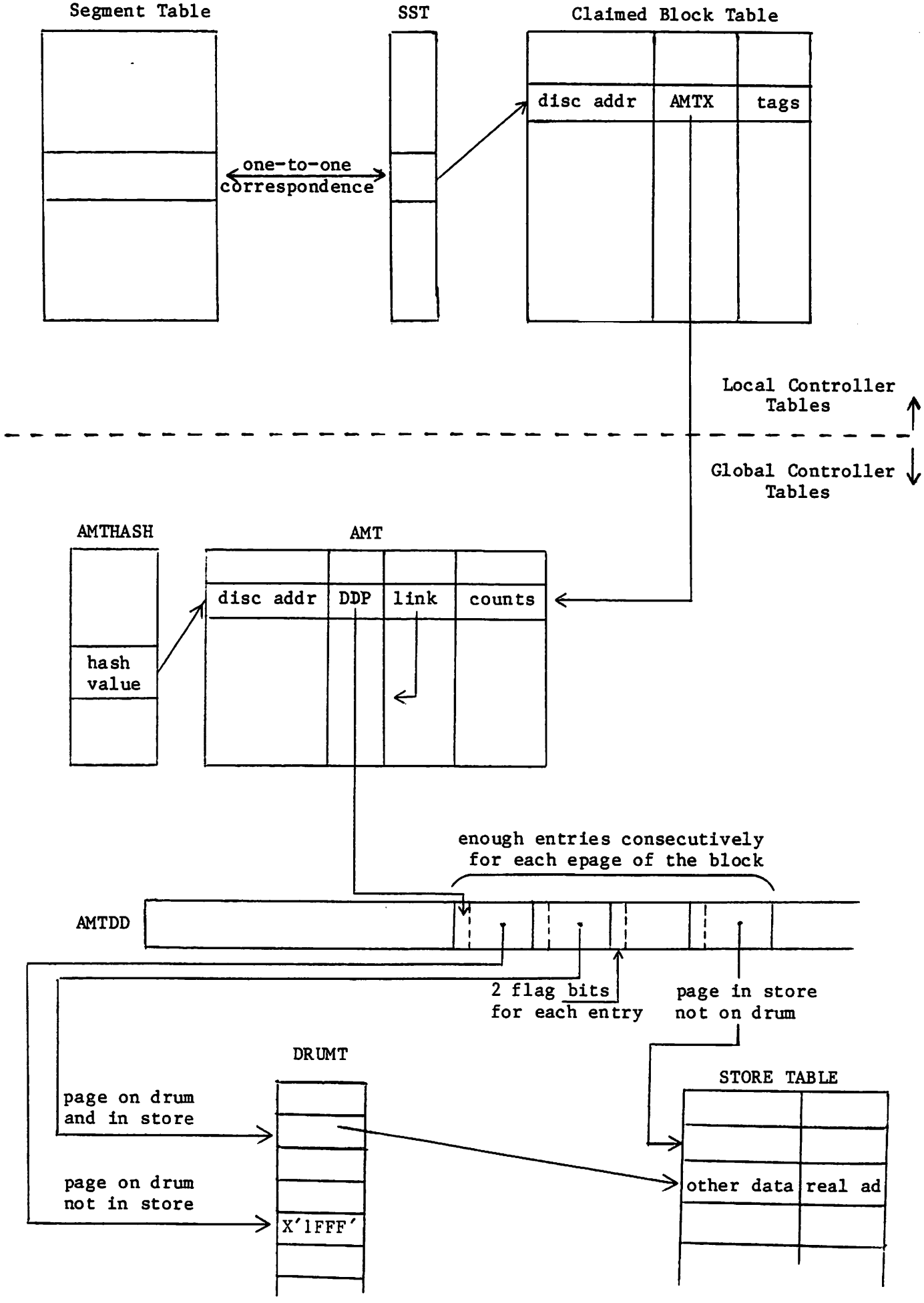
DA	disc address of block
DDP	index into AMTDD array for the start of block
LINK	link to other AMTs having the same disc address hash key
USERS	number of users of this block plus users of each page in this block
LEN	length of block
OUTS	number of page-outs in progress for this block

AMTDD halfintegerarray (0:extensible in epage units as required)

15	"new" epage
bit 14	0 Store Table Index follows 1 Drum Table Index follows
bits 13->0	Store or Drum Table Index (X'1FFF' indicates no store epage allocated)

DRUMT halfintegerarray (0:drum size -1)

bits 13->0 Store Table Index (X'1FFF' if unallocated)


Segment Table
SST
Claimed Block Table
disc addr
AMTX
tags
one-to-one
correspondence
Local Controller
Tables
Global Controller
Tables
AMTHASH
AMT
disc addr
DDP
link
counts
hash
value
enough entries consecutively
for each epage of the block
AMTDD
2 flag bits
for each entry
page in store
not on drum
DRUMT
STORE TABLE
page on drum
and in store
page on drum
not in store
X'1FFF'
other data
real ad

Note the one-to-one correspondence between DRUMT and logical drum epages such that the Drum Table Index doubles as the logical drum address. This precludes any "alternative track allocations" but this is unimportant: if the drum page is defective the information can be fetched from disc.

```
STORE    (byteinteger FLAGS, USERS, halfinteger, LINK, c
            BLINK, FLINK, integer REALAD)
```

(one entry per usable epage)

FLAGS	bits 7->6	11: disc write in progress
		10: disc read in progress
		00: no disc transfer in progress
	bits 5->4	11: drum write in progress
		10: drum read in progress
		00: no drum transfer in progress
	bit 3	"written to" since leaving drum or disc
	bit 2	0: page-out to disc only
		1: pageout to disc and drum
	bit 1	make "new" upon page-out
	bit 0	recapturable (when on free list)

USERS number of users of this page

LINK link to page-in-transit list (when a read is in progress only), or (when on free list and recapturable):

bit 15	0: AMTDD Index follows
	1: DRUMT Index follows
bit 14->0	AMTDD or DRUMT Index

BLINK back link to previous entry on free list; if page is in use then this is EPX, the page number in the block defined by AMTX

FLINK forward link to next entry on free list; if page is in use then this is AMTX

REALAD real store address of this epage

Page-in-transit Tables

These are lists of parameter cells containing reply parameters for all those processes awaiting the epage.

AMT and AMTDD

They are each mapped into their own (public) segments (21 and 22 respectively), so that they can be extended as required by appending an epage and adding the new space to the appropriate free lists.

Free lists

The tables in general have free lists linking unused entries:

CBT — Free entries are identified by being zero.

AMT — A pushdown list of free entries linked via the LINK field. The list head is AMTASL, in routine ACTIVE MEM. This table can be extended as necessary.

AMTDD — Separate pushdown free lists for each possible size of entry, i.e. 1-> MAXBLOCK epages. The list heads are kept in array DDASL(1:MAXBLOCK), in routine ACTIVE MEM. If a free list is empty a three-step procedure is used:

1) An attempt is made to fragment a larger entry.

2) All free lists are garbage-collected to recover any fragmentation losses.

3) The AMTDD is extended by adding an epage.

DRUMT — A pushdown list is maintained (by routine PAGETURN). The listhead is DDTASL, a Global Controller variable. A special arrangement is made for returning page-frames on which transfers have failed.

STORE — A complex doubly linked list (forward links in FLINK, back links in BLINK) is maintained to allow "recaptured" pages to be removed from the middle. Global heads FSTASL and BSTASL are used for removing and adding pages respectively. This arrangement maximises the chance of recapturing pages. Non-recapturable epages are added to the head of the list.

As well as the above, each Local Controller maintains an extract of the information for efficiency reasons. This information is held for a maximum of 32 distinct active segments. The figure of 32 (which is ample; System 4 used 24 much smaller segments) is so that bit arrays (0:31) can be kept in an integer.

AS — longintegerarray (0:31)

Contains a bit for each active epage in each active segment. This array is indexed from the primary segment table using bits 26-31 (marked as "usable by software" in PLI). Note that this assumes a maximum of 64 epages/segment, i.e. a minimum epage size of 4.

ASEG — byteintegerarray (0:31)

Contains the segment number of the active segment as a back pointer to SST when needed.

ASWAP — integer masquerading as bitarray (0:31)

Marks which entries in AS actually have an active page (i.e. have been referenced) in the current residence period.

ASWIP integer masquerading as bitarray (0:31)

Marks which entries in AS are used in which no page has yet been referenced in the current residence period.

ASFREE integer masquerading as bitarray (0:31)

Marks unused entries in AS. Note that ASFREE is always equivalent to (ASWAP!ASWIP)!!(-1).

OLDASWIPS integerarray (0: MAXRESIDENCES)

Contains ASWIP at the end of the previous MAXRESIDENCES residence periods and is used to de-activate segments as they move out of the working set.

These tables are used extensively in kernel operations such as paging and strobing. They are also used in the following non-kernel operations:

a) Connection

This operation consists of Director entering the block chains into the CBT for the disc blocks of the file being connected. This is followed by setting the SST entry, and the APF and length field for the segment(s) involved.

b) Activation

This is done on a block-by-block basis and involves obtaining AMT and AMTDD space and setting the pointers. Before the AMT space is obtained a search of the relevant hash chain is made. It is at this point that shared blocks are identified.

c) De-activation

De-activation is normally done on a segment basis, when examination of OLDASWIPS shows that the segment has been active for a period. Forced de-activation is also performed when there are 32 active segments, or at the Director's request before disconnection.

There are two parts to de-activation:

Step 1: Decrement the AMT_USER fields. Destroy CBT_AMX links and reset the active bit in CBT_TAGS.

Step 2: When AMT_USER = AMT_OUTS = 0, de-allocate AMT, AMTDD and any drum space.

Step 1 is done immediately. For unshared, unused segments Step 2 follows immediately. For used blocks a delay is required while transfers complete. For shared blocks, de-activation will not complete until the last user disconnects the shared file.

d) Disconnection
e) Disconnect and Destroy

Disconnection can only be carried out on blocks marked "inactivate" in the CBT; Director must check this. Otherwise, disconnection is the inverse of connection.

A block cannot be destroyed until both steps of de-activation have been completed - otherwise corruption of the file system would be possible, by AMT registering invalid "sharing" between the old and any possible new use of the disc block. A special Local Controller service (OUT 17) is provided so that Director can wait for transfers to complete.

P.D. Stephens

EMAS 2900 SUPERVISOR NOTE

No: 4 (revised)
Date: 21/4/81

Local Stacks

To accommodate extra local stacks three new OUTs are introduced: 12, 13 and 14. Local stacks will now be numbered 1, 2, 3, etc.

At present,

Stack number 1 is the normal Director/User stack, segment 4

Stack number 2 is the signal stack, segment 6

The nomination (see below) of stacks 1 and 2 is pre-defined. At present, one more stack is available for nomination (number 3), but this can be increased.

OUT 12 : Nominate stack

DIROUTP_P1 = stack number

DIROUTP_P2 = segment number

P1 must be within bounds (1:3)
Stack number P1 must not already be nominated
P2 must be within bounds (4:max segment)
P2 must be an even number
Segment P2+1 must not be connected

OUT 13 : Denominate stack

DIROUTP_P1 = stack number

P1 must be within bounds (1:3)
Stack number P1 must be nominated
Stack number P1 must not be the current stack

OUT 14 : Swop stack

DIROUTP_P1 = new stack number

P1 must be within bounds (1:3)
Stack number P1 must be nominated
Stack number P1 must not be the current stack
The new SSN+1 context is checked to ensure that:
* LNB and SF stack segments agree
* PSR_ACS is not zero

D.J. Rees

EMAS 2900 SUPERVISOR NOTE

No: 5
Date: 8/6/78

externalroutine GPC(recordname P) - Specification

1. Conventions

The record spec for the parameter is

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

The DEST and SRCE fields should be set up in the message to GPC as follows:

DEST considered as two halfwords DSNO and DACT, must have DSNO = GPC routine DSNO (picked up from an initialisation call to the originator), and DACT as appropriate

SRCE considered as two halfwords SSNO and SACT, must have
SSNO = SNO of originating routine, and
SACT = DACT-for-reply.

In addition, if GPC is being called or if a reply is not required, the top bit should be set in SRCE. For DACT= 4, 5 or 10 when GPC is called, the reply is the contents of the record P on return. If the request is being ponned to GPC, the top bit should not be set, and GPC will pon the reply.

2. Device entries

Each device on a GPC is allocated an area called a DEVICE ENTRY at grope time. The format of this area is given below. Fields marked * are intended to be used (read) by the calling routine; otherwise fields are either spare or for use by the GPC routine. The address of the DEVICE ENTRY for a given device is given by the GPC routine in P_P3 each time a message or reply is given to the relevant service routine, viz. in replies to ALLOCATE, DE-ALLOCATE, or EXECUTE CHAIN, in normal and abnormal TERMINATION responses, and in ATTENTION, RI and PCI responses. The format of device entry follows (all integers):

	SER	
	GPTSM	GPC-no, port, trunk, stream, mechanism
*	PROPADDR	pointer to property codes obtained at grope
	TICKS SINCE	for the time-out
*	CAA	comms area address
	GRCB AD	address of GPC's RCB
	LBA	
	ALA	
	STATE	

```
* RESP0          Response words from the most recent "user" request.
* RESP1

  SENSE1,SENSE2,
  SENSE3,SENSE4

  REPSNO

  BASE

* ID

  DLVN

* MNEMONIC       device mnemonic
```

Above this line, the format is common with that of externalroutine DISC.

```
  ENTSIZE        size of this whole entry in bytes

  PAW

  USAWO

  URCB AD        pointer to user's RCB (most recent request)

* SENSDAT AD     pointer to sense data obtained by routine GPC at most
                 recent abnormal termination for the device.

* LOGMASK        used to control GPC's logging of abnormal terminations
                 (may be altered by the relevant device service routine,
                 provided it has suitable acess to the segment). The
                 bottom byte of this word is considered as a mask for the
                 device secondary status. Following an abnormal
                 termination, if the AND of the mask and the secondary
                 status byte is non-zero, the device entry is dumped by
                 GPC on the main log.

* TRTAB AD       address of a translate table to turn characters which do
                 not exist in the repertoire into "delete" (EBCDIC 7) for
                 LP devices. (Translating with this table will have no
                 effect if CARTRIDGE is set to ZERO on the LP at grope
                 time.)

* UA SIZE        the size of the "user area" in bytes for devices to be
                 driven by ADAPTORS. These values are constants in the
                 GPC GROPE routine.

* UA AD          the address of the user area.

  TIMEOUT
```

The IMP recordformat statement defining this entry is available in file DIRARC.CFILE_ENTFORM on the Edinburgh EMAS 2900 system.

3. Initialisation

The first call of GPC must be the initialisation call (DACT=2), P1 = address of pre-prepared configuration table. DEST must contain the SNO for the GPC routine. GPC calls the following routines with a DACT of 4 for initialisation:

TAPE, OPER

4. Inputs to GPC

Apart from the initialisation call, the following DACT values are valid:

3 Interrupt from controller	P1= port+trunk (rightmost 2 quartets)
4 Allocate device	P1= device mnemonic P2= 0 CALL the service routine on interrupts. P2= 1 PON responses to the service routine.
5 De-allocate device	P1= device mnemonic
10 Execute chain	P1= RCB address P2= sno for device returned from Allocate P3= (PAW function)<<4 ! SAW flags P4= an IDENT field to be returned in P_P6 of TERMINATION and ATTENTION responses P5= 0 Normally P6= 0 Normally

Note that if P5 and P6 are not both zero, they will be taken to contain the Local Segment Table limit and base, and an ACR value, which will be written into the caller's RCB as pointed to by P_P1.

(This provides a mechanism for a privileged paged process to make direct requests on the GPC routine by using the DOUT18 Director procedure (Director Note no. 11), which ensures that a specified area of (local) virtual store resides in main store until a response is available.)

5. Responses to callers' requests

P1 = 0 means successful; otherwise:

4 Allocate 5 De-allocate	P1= 1 mnemonic invalid/not found P1= 2 already (or not) allocated

For a successful Allocate, P2 = identifier for device, to be supplied in P2 of Execute chain requests, P3 = address of the DEVICE ENTRY for the device.

10 Execute chain	P1= 1 device not found or not allocated to this service routine P1= 2 slot busy P3= address of the DEVICE ENTRY for the device

6. Action on stream response

The GPC routine response to an interrupt is to kick the appropriate service routine, which is called or ponned according to the P2 setting in the Allocate request. (Note that in the case of chains which terminate rapidly, for example faulty chains, the termination may be received before the GPC routine has returned to the caller, or ponned its acceptance of the request. In this case if responses to Execute chain requests are to cause a call of the service routine, this call would be recursive, and the replies may be received in the 'wrong' order.)

In the activation of the service routine, the record P is set as follows:

(a) For ATTENTION responses

DACT = 3

P1 = stream response word RESP0, byte 0 overwritten with device identifier as returned by Allocate. (Byte 2 is the attention byte. The RESP0 value is not written into the RESP0 field in the device entry for ATTENTION responses.)

P3 = address of the DEVICE ENTRY for the device.

(b) For NORMAL TERMINATION responses

DACT = 2

P1 = stream response word RESP0, BYTE 0 overwritten with device identifier as returned by Allocate. (Byte 2 is device primary status.)

P2 = stream response word RESP1.

P3 = address of the DEVICE ENTRY for the device.

The stream response words RESP0, RESP1 are also available in the corresponding fields in the DEVICE ENTRY.

(c) For ABNORMAL TERMINATION responses

DACT = 5

P1,P2 as for normal termination responses

P3 = address of the DEVICE ENTRY for the device

P4 = response analysis flags from the response to the SENSE command issued by GPC on behalf of the service routine. If this (right-most) byte does not indicate successful termination, then the sense information is suspect.

Status information for the device is pointed to by the field SENSAD in the DEVICE ENTRY. It is laid out as follows: secondary status followed by tertiary status (as many bytes as appropriate to the device), then primary status.

The stream response words RESP0, RESP1 are also available in the corresponding fields in the DEVICE ENTRY.

If a chain has "timed-out", due to having taken longer than a value set at grope-time for the relevant device-type, then a special abnormal termination response, in which RESP1=-1, is sent to the service routine.

(d) For PCI responses

DACT = 6

P1 = stream response word RESP0, byte 0 overwritten with the device identifier as returned by Allocate.

P3 = address of the DEVICE ENTRY for the device.

(e) For RI responses

DACT = 7

P1 = stream response word RESP0, byte 0 overwritten with device identifier as returned by Allocate.

P3 = address of the DEVICE ENTRY for the device.

J.K. Yarwood

No: 7
Date: 15/12/77

Elapsed Interval Timer

```
externalroutine ELAPSED INT (recordname P)

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

P_DEST X'A0000' + activity

activity 1 - Q/UNQ nominated routine for PON every n seconds.
2 - Q nominated routine for PON in n seconds.

P_P1 service number and activity of nominated routine.

P_P2 either: number of seconds to elapse before PON(0<P_P2<X8000)
or: if activity=1 and P_P2=-1 then unqueue nominated routine.

A reply will be PONed unless P_SRCE <= 0

where P_P1 = 0 if request successful
P_P1 ≠ 0 if routine already Q'ed/UNQ'ed or P_P2 invalid.

John Maddock

EMAS 2900 SUPERVISOR NOTE

No: 8
Date: 25/1/78

Delayed PON Facility

Although the ELAPSED INT facility allows service calls to be delayed, it only allows an activity number to be specified and not a whole parameter record. A new 'DPON' routine does allow this:

externalroutinespec DPON (recordname P, integer DELAY)

The effect is to copy the parameter record P into the normal PON/POFF mechanism parameter space but not to queue it. ELAPSED INT is called to PON a service ('DPONPUTONQ', service 12) in DELAY seconds, which then does put the parameter record on the appropriate queue, thus getting the normal PON effect.

The facility is available from SUP 16D.

D.J. Rees

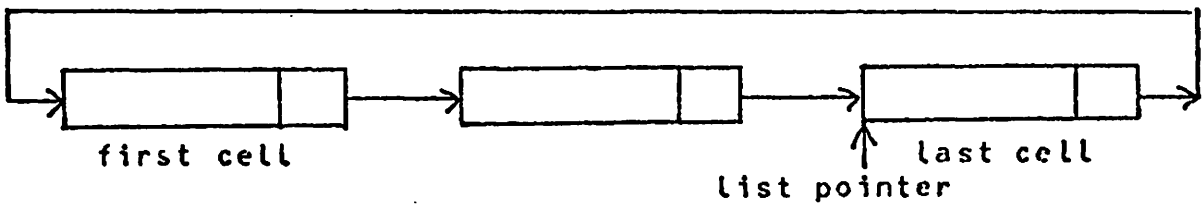
Parameter Passing Data Structures

```
%RECORDFORMAT PARMF(%INTEGER DEST,SRCE,P1,P2,P3,P4,P5,P6)
%EXTERNALRECORDARRAY PARM(1:parmn)(PARMF)
%EXTERNALINTEGERARRAY PARML(1:parmn)=2,3,4,.....
  ....,parmn,1
%EXTERNALINTEGER PARMASL=parmn

%RECORDFORMAT SERVF(%INTEGER P,L)
%EXTERNALRECORDARRAY SERVA(1:maxserv)
%EXTERNALINTEGER KERNELG,RUNQ1,RUNQ2=0
```

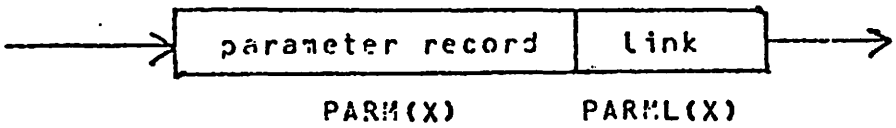
[Currently, parmn=256, maxserv=320]

All lists in this scheme are circular i.e. the link in the last cell points to the first cell. The list pointer always points to the last cell.

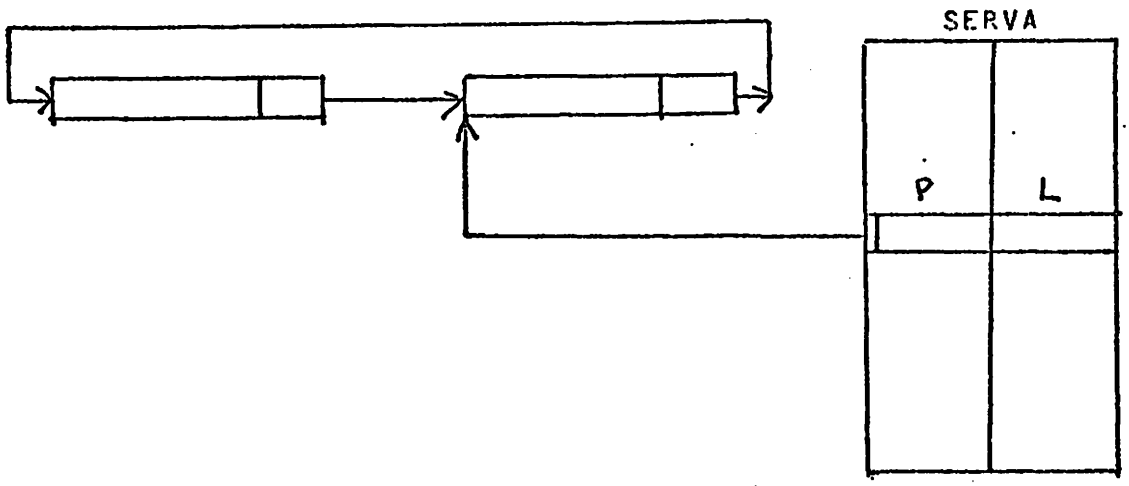


If the list contains only one cell then its link points to itself. If the list is empty the list pointer is zero.

There is a separate list for each service number which contains parameter records destined for that service. Each cell on the list consists of a pair from the parallel arrays "PARM" and "PARML".



The list pointers to these lists are held in the "SERVA" record array which is indexed by service number. Spare cells of this form are linked into an available space list with the list pointer "PARMASL".



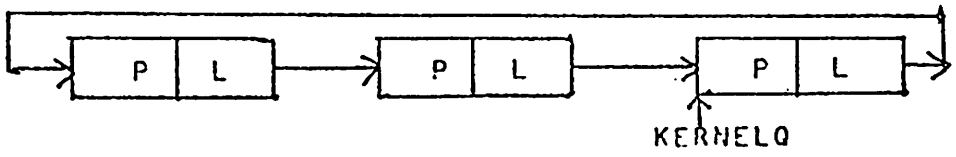
The list pointers occupy the bottom 31 bits of the record sub-element "P". The top bit i.e. the sign bit, is used to indicate the inhibit status of that service:

0 : uninhibited i.e. P>=0
1 : inhibited i.e. P<0

The "L" sub-element is used as a link for three lists which constitute a three priority-level CPU queue. The three lists, in order of priority, have pointers:

1. "KERNELQ"
2. "RUNQ1"
3. "RUNQ2"

For example, the queue of kernel services:



The "L" link values are the service numbers themselves.

D. J. Reed
17/2/78

EMAS2900 SUPERVISOR NOTE

No: 10
Date: 22/3/78

EMAS 2900: Sizes of resident code modules

1. Description of modules

Mnemonic	Description
GC	Global controller: stack-switching interrupt handler, real-time clock, main store and drum allocation low-level scheduling, process creation, hardware segment and page table maintenance, low-level resource-use monitoring.
LC	Local controller (an incarnation for each process): non-stack-switching interrupt handling (including virtual store interrupts and program errors), asynchronous message-handling for the process, timeslicing, communications message handling, process behaviour monitoring.
LC(Comms)	Communications local controller, plus LP and FE Adaptors.
MSG-SWITCH	System message switch-gear.
COMMAREA	System communication area module.
SFC	SFC (fixed-head disc) handler: queuing and execution of page-transfer requests.
DFC	DFC (removable disc) handler: queuing and execution of page-transfer and "private chain" requests, disc labelling and formatting.
GPC	GPC (tape and slow device) handler: queuing and execution of tape and slow device transfer requests (PRINT, OPER, TAPE and LINK modules formulate the requests).
OPER	OPER console handler.
PRINT	LP handler: for Supervisor log messages (these may alternatively be written to files on disc).
TAPE	Tape handler for Supervisor and paged processes.
DIAG	IMP language run-time diagnostic package for resident code.
STACK	Run-time processor stack

2. Code sizes (bytes)

Module	Code size		Work area	
	Decimal	Hexadecimal	Decimal	Hexadecimal
GC + LC	38528	X9680	6504	X1968
LC(Comms)	16312	X3FB8	7600	X1DB0
HSG-SWITCH	7160	X1BF8	12624	X3150
COMMAREA	9376	X24A0	632	X278
SFC	5856	X16E0	6936	X1B18
DFC	12184	X2F98	1024	X400
GPC	5960	X1748	592	X250
OPER	5864	X16E8	1464	X5B3
PRINT	10560	X2940	704	X2C0
TAPE	12280	X2FF8	1104	X450
DIAG	4824	X12D8	248	XF8
STACK	-	-	16384	X4000
Totals:	128904	X1F788	55816	XDA08

Total resident code + work area: 184,720 X2D190

J.K. Yarwood

2950: %EXTERNALROUTINE DCU (%RECORDNAME P) - Specification

1. Conventions

The record spec for the parameter is

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

The DEST and SRCE fields should be set up in the message to DCU as follows:

DEST considered as two halfwords DSNO and DACT, must have DSNO = DCU routine DSNO (picked up from an initialisation call to the originator), and DACT as appropriate

SRCE considered as two halfwords SSNO and SACT, must have
SSNO = SNO of originating routine, and
SACT = DACT-for-reply.

In addition, if DCU is being called or if a reply is not required, the top bit should be set in SRCE. For DACT= 4, 5 or 10 when DCU is called, the reply is the contents of the record P on return. If the request is being ponned to DCU, the top bit should not be set, and DCU will pon the reply.

2. Device entries

Each device on a DCU is allocated an area called a DEVICE ENTRY at grope time. The format of this area is given below. Fields marked * are intended to be used (read) by the calling routine; otherwise fields are either spare or for use by the DCU routine. The address of the DEVICE ENTRY for a given device is given by the DCU routine in P_P3 each time a message or reply is given to the relevant service routine, viz. in replies to ALLOCATE, DE-ALLOCATE, or EXECUTE CHAIN, in normal and abnormal TERMINATION responses, and in ATTENTION and PCI responses. The format of device entry follows (all integers):

	SER	
	PTSM	DCU-no, stream, mechanism
*	PROPADDR	pointer to property codes obtained at grope
	SECS SINCE	for the time-out
*	CAA	comms area address (see below)
	GRCB AD	address of Transfer Control Block (TCB) used by DCU code
	LAST ATTN	last attention byte obtained from device
	ALA	not used
	STATE	

	RESP0	
	RESP1	
	SENSE1,SENSE2, SENSE3,SENSE4	Software will read Sense data into the postamble of the user's TCB, if there is one, or into these four words otherwise. This facility may be withdrawn when postambles are handled properly by the DCU. For the time being, the sense data should be found by using the SENSDAT AD field (see below).
	REPSNO	
	BASE	
*	ID	Copy of P4 from parameter in last 'execute chain' command
	DLVN	
*	MNEMONIC	device mnemonic

Above this line, the format is common with that of externalroutine DISC.

	ENTSIZE	size of this whole entry in bytes
	PAW	
	USAW0	
	URCB AD	pointer to user's first TCB (most recent request)
*	SENSDAT AD	pointer to sense data obtained by routine DCU at most recent abnormal termination for the device.
*	LOGMASK	
*	TRTAB AD	address of a translate table to turn characters which do not exist in the repertoire into "delete" (EBCDIC 7) for LP devices. (Translating with this table will have no effect if CARTRIDGE is set to ZERO on the LP at grope time.)
*	UA SIZE	the size of the "user area" in bytes. These values are constants in the DCU GROPE routine. TCBs for the device must be held in this area.
*	UA AD	the address of the user area.
	TIMEOUT	
	PROPS0	
	PROPS1	

The IMP recordformat statement defining this entry is available in file DIRARC.CFILE_DCUENTF on the Edinburgh EMAS 2900 system.

3. Initialisation

The first call of DCU must be the initialisation call (DACT=2), P1 = address of pre-prepared configuration table. DEST must contain the SNO for the DCU routine. DCU calls the following routines with a DACT of 4 for initialisation:

TAPE, OPER, MK 1 FE ADAPTOR

and DCU also PONs a request to the interval timer to get itself kicked every few seconds.

4. Inputs to DCU

Apart from the initialisation call, the following DACT values are valid:

3	Interrupt from controller	P1= interrupt word
4	Allocate device	P1= device mnemonic P2= 0 CALL the service routine on interrupts. P2# 0 PON responses to the service routine. P2 is ignored for 'non-recognised' routines, to which responses will always be PONned.
5	De-allocate device	P1= device mnemonic
6	Clock interrupt	(all parameters ignored)
8	Ignored	
10	Execute chain	P1= address of caller's first TCB P2= sno for device returned from Allocate P3 is ignored P4= an IDENT field to be returned in P_P6 of TERMINATION and ATTENTION responses

5. Parameters and responses

DCU will only accept calls with DACT= 4, 5 or 10 (allocate, deallocate, execute chain) from "recognised routines". The acceptable service numbers (decimal) are 49, 50, 54, 57 and the routines are assumed to be TAPE, OPER, PRINTER, MK1FEADAPTOR respectively. These are the routines which are called to notify events which occur on devices allocated to routines with the appropriate service numbers (if direct calling rather than PONning was requested when the device was allocated). Reliable routines can be added to the list on request, by modifying the source and recompiling.

DCU will accept no calls before an initialise call (DACT=2). It does not check the SRCE number for DACT=2 (initialise), 6 (clock interrupt) or 3 (DCU interrupt). It could easily be made to do so.

When DACT= 2, 3 or 6, the value of the parameter on exit is undefined, and no reply will be PONned under any circumstances (although other routines may be called or messages PONned to them).

For DACT= 2 or 6, the value of the parameter on entry is immaterial. For DACT=3, P1 must have a copy of the DCU interrupt word.

For all calls on DCU, the reply will be found in P on exit (having overwritten the original value). For DACT = 4 (allocate), 5 (de-allocate) or 10 (execute chain), a copy of the reply will also be PONned to the caller. This is the only way that a response can get back to the caller if he PONned his original message to DCU instead of calling DCU. If he did call DCU, then the PONned reply is clearly redundant, and it can be suppressed by setting the most significant bit of SRCE before calling DCU. If that bit is set in a message PONned to DCU, then no reply will ever get back to the caller.

For other values of DACT, no reply is necessary. Nothing will be PONned to the caller, regardless of the setting of the top bit of SRCE. If DCU has been called, then P may have been overwritten, but its new contents are not specified here.

Values not specified in the responses from DCU are undefined. They need not be the same as the values supplied by the caller.

DACT=4, ALLOCATE P1 mnemonic

P2 =0: "call me to notify events" - only works for recognised routines
#0: "PON notification of events to me"

N.B. P2 does not affect the reply from the present allocate request; whether that is PONned or not depends on the top bit of SRCE. Nor does P2 affect the reply to any other call or PONned request to DCU. P2 controls how the user will be notified of interesting events which are not immediate responses to calls or PONned requests; for example, the termination of a transfer.

Response P1=1 unrecognised DACT no or mnemonic not found
2 device already allocated
3 caller not acceptable
0 O.K.

if P1=0: P2 device service no, needed for subsequent use of the device
P3 address of the device entry (q.v.)
P4 0, or -1 if device busy
P5
P6 mnemonic of device allocated. You need this to deallocate the line printer, if you asked for 'LP', because deallocate won't accept 'LP'.

But, for opers:
P3 address of buffer area (or 1's complement of no. of screens if device busy)
P4 no of screens
P5 size of buffer area

Note that P4<0 means that the device is still active: a response is expected, and execute chain requests will not be accepted until the response has arrived.

DACT=7 special allocate
P3 = increment & addr & FAT accts
P4 = service for attns (in bottom of word!)

DACT=5, DE-ALLOCATE

P1 mnemonic

You may de-allocate a device which is allocated to someone else (DCU could check that easily), or a device which is busy.

Response

P1 = 0 O.K.
1 unrecognised DACT number
2 device not allocated
3 caller not acceptable
P2 = 1 mnemonic not found ('LP' will give this result, although allocate will accept 'LP')
P3 = address of device entry (why?)

DACT=8 IGNORED

On exit the parameter P will be undefined. No reply will be PONned under any circumstances.

DACT=10, EXECUTE CHAIN

P1 address of caller's first TCB
P2 service number for device (as returned by allocate)
P4 copied to the ID field in the device entry and returned to the caller with notification of termination of transfer.

The user must clear the response word to zero in each TCB before calling execute chain.

Response

P1 = 1 unrecognised DACT number or unrecognised service number
2 slot already on queue
3 source not acceptable
P3 = address of device entry

6. Action on stream response

The DCU routine response to an interrupt is to kick the appropriate service routine, which is called or ponned according to the P2 setting in the Allocate request. (Note that in the case of chains which terminate rapidly, for example faulty chains, the termination may be received before the DCU routine has returned to the caller, or ponned its acceptance of the request. In this case, if responses to Execute chain requests are to cause a call of the service routine, this call would be recursive, and the replies may be received in the 'wrong' order.)

In the activation of the service routine, the record P is set as follows:

(a) For ATTENTION responses

top byte P1 = device ID as allocated
P3 = address of slot

DACT = 3

P1: byte 2 is the attention byte
P3: address of slot

(b) For NORMAL TERMINATION responses

DACT = 2

P5 ~~P1~~ = address of last TCB executed in the chain
P6 = P-P4 in last create

(c) For ABNORMAL TERMINATION responses

DACT = 5

P2 -1: timed out
0: see sense bytes
+1: DCU failure
8: sense bytes lost

P5 = address of last TCB executed in the chain.
P6 = P-P4 in last execute chain
Status information for the device is pointed to by the field SENSAD in the DEVICE ENTRY. It is laid out as follows: secondary status followed by tertiary status. If the relevant TCB had 'postamble valid' set, the sense bytes will be in the postamble; otherwise they will be in the appropriate fields in the device entry. When DCU hardware supports postambles, the software facility to put sense data in the device entry may be withdrawn.

If a chain has "timed-out", due to having taken longer than a value set at grope-time for the relevant device-type, then a special abnormal termination response, in which P2=-1, is sent to the service routine.

(d) For PCI responses

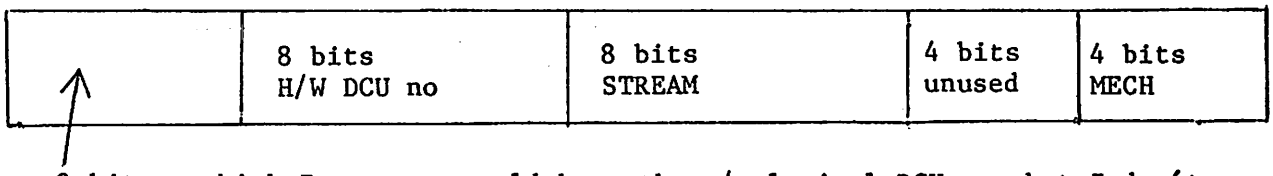
DACT = 6

P5 = address of a TCB which is later in the chain than the one whose termination caused PCI.

7. Major changes between GPC and DCU

1. DCU uses a pseudo comms area in which the first word has the image store address of the interrupt/activate words of the DCU. The rest of the area, if any, is used for work space for opers, as defined by the oper start and limit fields in the configuration table.

2. The word PTSM in the device slot is laid out like this:



8 bits - which I suppose could have the s/w logical DCU no, but I don't think this field is ever used in my code so it probably doesn't matter if you put something else in there.

3. The table PTS TO SLOT is replaced by a table DCU AND STRM TO SLOT which is laid out in order of logical dcu numbers, with 256 one-byte entries for the streams for each DCU.

4. DCU will not accept calls from anyone except the 'recognised service routines'. Other calls are rejected with P1=3. Users must agree on some segment for TCBs and the DCU must be initialised to use that.

5. DCU requires the caller to set up a chain of TCBs, instead of the RCB and logic block and address list of the GPC. The worst problem here is that each TCB must contain one word which is derived from the segment table entry for the segment containing the data. Also all TCBs must be in the same segment, so when you get a device allocated you are given an area (indicated by the fields UA SIZE and UA AD in the device entry) in which you must construct all your TCBs. Users must clear the response word to zero in each TCB before calling "execute chain".

J. Wexler

EMAS 2900 SUPERVISOR NOTE

No: 14
Date: 22/1/79

Tables Used by the Drum Driver

Two tables are set up by grope for routine DRUM.

The first is the DRUM CONTROLLER TABLE and is to be found at address COM_SFC TAD. There is one 6-word entry for each controller, corresponding to the following format:

```
recordformat CONTABF(integer ISCONTREG, BATCH, integername MARK, CRESPO)
```

where:

ISCONTREG is the image store address to send channel flags to the controller.

BATCH is a count of the streams for which activations have been set but not fixed.

MARK is a reference to the first word of the communication area.

CRESPO is a reference to the controller response word for the SFC.

The second table is the drum table, containing one entry per drum. Different sizes of drum can be attached to the same SFC (!), so this table must contain all relevant information. Drums actually work with 1 Kilobyte sectors but the software makes them appear to work in Extended Sectors (ESECS), which are the same size as an EPAGE. If an integral number of ESECS does not fit on a track the remnant is discarded.

The format of a drum table entry is:

```
recordformat DTABF (integer NSECS, halfinteger LOGI, CONTI, integer c
                    SECLIM, NEXT, STATE, integername MARK, PAW, PIW, c
                    record ESCB0, ...., ESCBN (ESCBF))
```

where:

NSECS is the number of 1K sectors on the drum, ignoring any discarded because the track size is not an exact multiple of epage size.

LOGI is an index into an error logging table

CONTI is an index to the controller table for the controller of this drum.

SECLIM is the number of 1K sectors per track (again ignoring any discarded).

NEXT is the address of the next entry in the drum table (needed as this entry is of variable length).

STATE is the drum state - the sign bit set indicates an inoperable drum; otherwise a small integer.

MARK
PAW
PIW — these are references to the appropriate words in the communication area.

ESCB0 - ESCBN — extended sector control blocks - one for each extended sector; the number present = SECLIM/epage size. These control the queuing of transfers for each extended sector.

Each extended sector control block has the following format:

recordformat ESCBF(integer HQ, LQ, SAW0, PAWBITS, ADDSTRMS)

where:

HQ, LQ — are the heads of queues of high and low priority transfers. The sign bit is set when the top transfer is fired.

SAW0 — is the stream activation word for the first (1K) sector of this extended sector. Bit patterns for the remaining 1K sectors are generated from this one.

PAWBITS — A pattern of bits that, when ORed into the primary activation word, will start all 1K sectors of this extended sector transferring.

ADD STRMS — The address of the stream block (4 words: SAW0, SAW1, RESP0, RESP1) for the first 1K sector of this extended sector.

The queue of transfers pending is kept in normal parameter passing cells. The seven words are used as follows:

recordformat ESQF(integer RDEST, RSRCE, IDENT, ESECNO, STORI, ADPTS, c
longinteger LSAW, integer LINK)

where:

RDEST, RSRCE — are the destination and source for replying.

IDENT — is the returnable identifier provided with the request.

ESECNO — is the drum page - all drums being treated as one logical array of extended sectors.

STORI — is the index to the store array which identifies the core epage from which (or to which) the transfer is to take place.

ADPTS — = 0 for reads
= address of a count to be decremented on completion for writes.

LSAW — secondary activation word for first 1K drum sector of the extended sector.

LINK — to the PP cell containing the next transfer.

Notes

* The first 6 words are the same as the request, except that DEST and SRCE have been switched.

* The queues are circular - the header points to the last cell which points to the first. A pushdown LIFO arrangement will be tested shortly.

P.D. Stephens

Communications Record Format

The record format described below is extant from Chopsupe 186 onwards:

```
recordformat COMF(integer OCPTYPE, IPLDEV, SBLKS, SEPGS, NDISCS, c
        DDTADDR, GPCTABSIZE, GPCA, SFCTABSIZE, SFCA, SFCK, DIRSITE, c
        DCODEDA, SUPLVN, WASKLOKCORRECT, DATE0, DATE1, DATE2, c
        TIME0, TIME1, TIME2, EPAGESIZE, USERS, PROCMON, DQADDR, c
        SACPORT, OCPPORT, ITINT, CONTYPEA, GPCCONFA, FPCCONFA, SFCCONFA, c
        BLKADDR, DPTADDR, SMACS, TRANS, longinteger KMON, c
        integer DITADDR, SMACPOS, SUPVSN, PSTVA, SECSFRMN, SECSTOCD, c
        SYNC1DEST, SYNC2DEST, ASYNCDEST, MAXPROCS, INSPERSEC, ELAPHEAD, c
        COMMSRECA, STOREAAD, PROCAAD, SFCCTAD, DRUMTAD, SP0, SP1, SP2,c
        SP3, SP4, SP5, SP6, SP7, SP8, SP9, LSTL, LSTB, PSTL, PSTB, c
        HKEYS, HOOT, SIM, CLKX, CLKY, CLKZ, HBIT, SLAVEOFF, INHSSR; c
        SDR1, SDR2, SDR3, SDR4, SESR, HOFFBIT, S2, S3, S4, END)
```

This format describes "The Communication Record", which is kept locked in store at Public address X'80C00000'. It is readable at all ACR levels but writeable at ACR 1 only. Its purpose is to describe the hardware on which the EMAS 2900 System is running. Each entry is now described in more detail:

OCPTYPE The 2900 Processor on this configuration, as follows:
1 = 2950 or S1
2 = 2960 or P2
3 = 2970 or P3
4 = 2980 or P4
6 = 2976 or P4/1

IPLDEV The port/trunk/stream (or DCU/stream) of the device used at IPL time. (X'FFF' used for tape IPLS.)

SBLKS The no of 128K blocks of main store present.

SEPGS The no of extended pages for paging (i.e. not including any pages occupied by resident code & data).

NDISCS Then number of EDS drives available.

DDTADDR The address of the Disc Device Table which has NDISC entries, currently of 88 bytes each.

GPCTABSIZE The size in bytes of the GPC (or DCU) table.

GPCA The address of the GPC (or DCU) table.

SFCTABSIZE The size of the SFC (i.e. DRUM) table.

SFCA The address of the SFC table.

SFCK The number of (useable) 1K page frames of Drum store available for paging. (0 = No drum configuration)

DIRSITE — The Director site address (e.g. X'200') - no longer required?

DCODEDA — The Disc Address of the Director (expressed as SUPLVN<<24!DIRSITE).

SUPLVN — The logical volume no of the disc from which the Sytem was "SLOADED". Various System components (e.g. DIRECT, VOLUMS) will page from here.

KLOKCORRECT — No longer used.

DATE0, DATE1, DATE2 — These three integers define the current date (updated at 2400) as a character string such that the length is in the least significant byte of DATE0.

TIME0, TIME1, TIME2 — These three integers define the clock time as a string in the same format as for DATE. The time is updated about every 2 seconds.

EPAGESIZE — The number of 1K pages combined to make up the logical "Extended Page" used in EMAS 2900. Currently=4.

USERS — The number of user processes (foreground+background) currently in existence. Includes DIRECT, VOLUMS and SPOOLR.

PROCMON — Was used for process monitoring. Now redundant.

DQADDR — The address of a public segment holding the header blocks for the disc transfer queues.

SACPORT — The bottom 16 bits hold the Port no of the Store Access Controller used to IPL the System. In dual SAC systems the other SAC is in the top 16 bits.

OCPPORT — The bottom 16 bits hold the Port no of the OCP used in the IPL. Like SACPORT, the top 16 bits will hold the Port of second OCP when duals are supported.

ITINT — The Interval Timer interval in microsecs. Varies between different members of the range.

CONTYPEA — The address of a 16 byte area containing the codes of the controllers in trunk order. Codes are:
0 = Not relevant to EMAS 2900
1 = SFC1
2 = FPC2
3 = GPC1

GPCCONFA, FPCCONFA, SFCCONFA — These three variables each point to a word array containing controller data. The first word in each case says how many controllers on the system. The remainder have Port&Trunk in top byte and Public segment no of comms segment in bottom byte. For GPCS the Public Seg no is apparently omitted!

BLKADDR — The address of first element of a word array bounds (1:SBLKS) containing the real address of each 128K block of main store. Real addresses are in the form RSN/SMAC NO/Address in SMAC.

DPTADDR — The address of the Disc Property Table. Now not needed as the DDT has addresses, not offsets, for properties.

SMACS — A bit mask of SMACS found at Grope time: 2**0 bit set if SMAC 0 found etc.

TRANS — The address of a 512 byte area containing 2 translate tables. The first is ISO to EBCDIC, the second the exact converse.

KMON — A 64 bit bitmask controlling monitoring of Kernel services. Bit 2**n means monitor service n. Bits can be set by Operator command KMON.

DITADDR — Disc index table address. The address of the first element of an array (1:NDISCS) containing the address of the disc device entries. Needed for S series and provided for compatablity on P series.

SMACPOS — The no of places that the SMAC no must be left shifted to be in the right position to access a SMAC image store location. Incredibly this varies between the 2980 and others!!

SUPVSN — The Supervisor identification number, as a three character string, e.g. 22A.

PSTVA — The virtual address of the Public Segment table which is itself a Public segment. All other information about PST can be found by looking at its own PST entry.

SECSFRMN — The no of seconds since midnight. Updated as for TIME.

SECSTOCD — The number of seconds to System closedown if positive If zero or negative no close down time has yet been notified. Updated as for TIME.

SYNC1DEST
SYNC2DEST
ASYNCDEST — These are the service nos N2,N3 & N4 for process parameter passing, as described in Supervisor Note 1.

MAXPROCS — The maximum number of paged processes that the Supervisor is configured to run. Also the size of the Process array.

INSPERSECS — The number of instructions the OCP executes in 1 second divided by 1000 (approx average for EMAS 2900).

ELAPHEAD — The head of a linked list of param cells holding service with an elapsed interval interrupt request outstanding.

COMMSRECA — The address of an area containing details of the Communication streams. (private to COMMS Control)

STOREAAD — The address of first element of the store record array bounds (0:SEPGS-1).

PROCAAD — The address of first element of the process record array bounds (0:MAXPROCS).

SFCCTAB, DRUMTAD — The addresses of two private tables provided by grope for use by the routine DRUM. They give details of the SFCS and DRUMS found on the system.

SP0->SP9 — Spare locations.

LSTL, LSTB, PSTL, PSTB, HKEYS, HOOT, SIM, CLKX, CLKY, CLKZ — These are the image store addresses for the following control registers:

- Local Segment Table Limit & Base
- Public Segment Table Limit & Base
- Handkeys, Hooter, System Interrupt Mask Register
- and the clock X,Y & Z Registers

HBIT — A bit pattern that when ORed into Control Register "HOOT" operates the Hooter. (0=Hooterless machine)

SLAVEOFF — A bit pattern (top 16 bits) and Image store address in bottom 16 bits. ORing the top 16 bits (after shifting) into the image store will stop all slaving of operands but not instructions.

INHSSR — A bit pattern and image location as for SLAVEOFF. ORing the bits into the location will switch off reporting of successful system retry.

SDR1, SDR2, SDR3, SDR4, SESR — The image store addresses of SMAC internal registers needed by the Engineers after SMAC errors have occurred.

HOFFBIT — A bit pattern that when ORed into a SMAC Engineers status register will stop reporting of errors from that SMAC.

P.D. Stephens

No: 16
Date: 21/11/79

GPC Device Entry Format

```
%RECORDFORMAT DEVICE ENTRY F(%INTEGER %C
     X1, %C
     GPTSM, %C
     PROP A, %C
     SECS SINCE, %C
     CA A, %C
     G RCB A, %C
     LB A, %C
     AL A, %C
     X2, %C
     RESP0, RESP1, %C
     SENSE1, SENSE2, SENSE3, SENSE4, %C
     X3, X4, %C
     IDENT, %C
     X5, %C
     MNEMONIC, %C
     DEVICE ENTRY S, %C
     PAW, %C
     U SAW 0, %C
     U RCB A, %C
     SENSE DATA A, %C
     LOG MASK, %C
     TR TABLE A, %C
     UA S, %C
     UA A, %C
     TIMEOUT, %C
     PROPS0, PROPS1)
```

where:

X1, X2, etc.	are spare.
GPTSM	GPC no, Port, Trunk, Stream, Mechanism.
PROP A	address of property codes obtained by GROPE.
SECS SINCE	time since current I/O request fired.
CA A	Communications Area address.
G RCB A	address of GPC's RCB.
LB A	
AL A	
RESP0, RESP1	response words set by GPC for most recent request. RESP1 = -1 after a time out.
SENSE1, 2, 3, 4	
IDENT	set by GPC from P4 of EXECUTE call.

MNEMONIC	device mnemonic.
DEVICE ENTRY S	size in bytes.
PAW	set by GPC from P3 of EXECUTE call.
U SAW 0	ditto.
U RCB A	set by GPC from P1 of EXECUTE call.
SENSE DATA A	address of sense data obtained by GPC after the most recent abnormal termination.
LOG MASK	used to control GPC's logging of abnormal terminations. Following an abnormal termination: 1. %UNLESS D_LOGMASK & byteinteger(addr(D_SENSE1)) = 0 %THEN 'dump the device entry to the main log'. 2. %UNLESS D_LOGMASK >> 8 = 0 %THEN 'read streams controller status'.
TR TABLE A	for an LP, the address of a translation table which turns characters which do not exist in the repertoire and which are not control characters into 'delete' (X'07'). This character is ignored by the printer. If the 'CART' on the printer is set to zero at initialisation time, translation has no effect i.e. f(i) = i.
UA S, UA A	the size in bytes and the address of a user area. Set up at GROPE time.
TIMEOUT	the time for which an I/O request is allowed to run before being failed 'timeout' by GPC.
PROPS0, 1	the device properties as obtained by GROPE.

A. Gibbons

No: 17
Date: 17/12/79

Basic Magnetic Tape Services

Services are offered on the relevant service number as supplied by Volumes.

Construct a record of the following format:

```
(integer DEST, SRCE, IDEN, halfinteger LENGTH, byteinteger c
 CONTROL, TYPE, integer ADR, LEN2, LSL, LSTBA)
```

where the fields have the following meanings:

DEST — X'310000' ! service number

SRCE — filled in by message-passing interface

IDEN — a private identifier

LENGTH — A signed number whose use is dependent on TYPE. If LENGTH=0 then the value in LEN2 is used in its place.

CONTROL — The individual bits may be interpreted as follows:

2**0 - Treat TM as block
2**1 - Suppress error retry
2**2 - Suppress short block checking
2**3 - Suppress long block checking

TYPE — Permissible values are:

0 - Erase one block
1 - read one block
2 - Write one block
4 - Check Read
5 - Private chain
8 - Skip a number of blocks
9 - Skip a number of tape marks
10 - Write a tape mark
17 - Rewind to BT

ADR — The address to be used in any data transfer.

LEN2 — May be used instead of LENGTH, especially where a size of greater than 32 Kbytes is required.

LSL — Local segment limit. This field and the next are only required if the user is specifying a private segment table to be used in the transfer of data (e.g. DOUT18). Note that this is always required where the data length exceeds 4 Kbytes. The bottom byte should be filled in with the user's ACR.

LSTBA — Local segment table base address.

The reply to a request is made in the following format:

(integer DEST, SRCE, IDEN, FLAG, P3, P4, P5, P6)

where the fields have the following meanings:

DEST — Originator of request

SRCE — request DEST field

IDEN — request IDEN field

FLAG — may have the following values:

0 - operation successful
1 - operation failed
2 - request rejected by TAPE routine
4 - unexpected TM, BT or ET, but otherwise successful

P3 — a value whose meaning is dependent on the request TYPE

P4 — Indicates when a short block or long block condition has arisen in an otherwise successful read request:

1 - short block
2 - long block

Details of action taken

ERASE (TYPE = 0)

A length of tape equivalent to a block of LENGTH bytes is erased.

READ (TYPE = 1)

A read to the given ADR is performed. The default response to failure is to attempt up to fifteen retries, reading in alternate directions.

If the length specified is negative then a backwards read is performed. The CONTROL bits may be used; these stop error retry and recovery from short or long block failures. If ADR is a local address then LSL and LSTBA are used to define a local segment table. The field P3 in the reply record will contain the number of bytes transferred, after a successful request.

WRITE (TYPE = 2)

A write from the given ADR is performed. The default response to error is to attempt up to five retries.

CHECK READ (TYPE = 4)

Similar to READ, but no data is transferred.

PRIVATE CHAIN (TYPE = 5)

See 'Private Chains', below.

FILE POSITION (TYPE = 8)

LENGTH blocks are skipped. If LENGTH is a negative number then the direction is backwards, otherwise forwards. Encountering a tape mark will normally cause a FLAG = 4. Note that the tape is positioned before the TM. Thus in normal mode it is impossible to exit a file using this command. However if CONTROL bit 2**0 is set then tape marks are treated as blocks. Field P3 in the reply is set equal to the number of blocks and tape marks skipped.

TAPE POSITION (TYPE = 9)

LENGTH x CONTROL tape marks are skipped, where conventionally; LENGTH specifies the number of files to be skipped, and CONTROL specifies the number of tape marks in each file.

A backwards skip is indicated by a negative LENGTH and may stop with FLAG = 4, indicating BT. A forwards skip will stop with FLAG = 4 either if ET is encountered or the first block in a file is a tape mark. In the second case the tape is left positioned between the tape marks.

Thus if CONTROL = 1 and LENGTH is large, the tape will be left at the first double tape mark found. Field P3 in the reply is set equal to the number of tape marks skipped.

WRITE TM (TYPE = 10)

One tape mark is written to the tape.

REWIND (TYPE = 17)

The tape is rewound to BT.

Queuing of Requests

Normally a user will wait for the reply to his current request before sending the next. However, some gain in speed can be made by sending a number of requests together. The first received will be actioned. The subsequent requests will be queued. As soon as the device terminates its first command another is released from the queue. Replies are sent normally and care may be required in the handling of errors, since execution of queued requests will continue regardless of the success or failure of prior commands.

Private Chains

This facility allows the user to specify his tape commands at a much more detailed level. The TAPE routine retains control of the deck, but passes the user's request directly to GPC. Private chain requests can be interspersed freely with the other commands documented here.

The user must provide:

a) A request control block (RCB)
b) A chain of logic block entries (LBEs)
c) A group of address list entries (ALEs)

The request format is:

DEST	X'310000' ! SNO
SRCE	0
IDEN	User-specified
P2	5
P3	Address of RCB
P4	A User service number if ATTNS are to be notified; otherwise 0
P5	0
P6	0

Information on RCBs, LBEs and ALEs should be sought in 'System Technical Description of GPC1' and the Engineer's manual for the tape deck concerned.

All data areas concerned, i.e. RCB, LBE, ALE and data buffers, must be 'locked down' before sending tbe request. This is achieved by use of DLOCK. They should be freed after the request has terminated by use of DUNLOCK.

Note that an ACR of 5 or lower is required to use these procedures.

The reply format is:

For Terminations:

P1	IDEN
P2	STREAM RESPONSE WORD 0
P3	(STREAM RESPONSE WORD 1<<14) ! Secondary Status Byte
P4 onwards	12 bytes of tertiary status information

For Attentions (usually obtained by use of DPOFF):

DEST	P4 from request (usually SYNC1 - i.e. UINFI(7))
P1	stream response word 0

Notes

1) The stream response word 0's have the SNO set in their top byte.

2) ATTNS will continue to be directed to the user until a non-private chain command is issued.

Modes of Operation

Eventually most interfaces by which a user can claim a tape will accept a MODE BYTE, to be passed on to the Supervisor tape routine. The format of this byte is:

Whole byte zero: 1600 BPI Phase Encoded (default)

bit 0 — 1 - interpret other bits

bit 3 — 0 - odd parity
1 - even parity

bit 5 — 1 - compress-expand (1900 mode)

bits 6&7

bit 6	bit 7	
1	1	1200 bpi (PE)
1	0	800 bpi (NRZI)
0	1	556 bpi (NRZI)
0	0	200 bpi (NRZI)

A. Anderson

TAPE VERSION 62 RELEASE NOTE

(ERCS04.TAPE62S, TAPE62Y) - MUST COMPILE PARMX.

CHANGES

1) NRZ 800 B.P.I. MODE SET IF 'CONTROL' TOP BIT IS SET. MODE RETURNS TO 1600 BPI P.E. WHEN TAPE IS RELEASED.

2) TAPE MARKS - COUNTED ON TAPE POSITION COMMANDS AND THE VALUE RETURNED IN REP_P3

3) CHECK READ - READ WITH NO DATA TRANSFER IMPLEMENTED (ERROR RECOVERY AS FOR NORMAL READ)

4) ANY SIZE OF LABEL NOW ACCEPTED. ONLY REQUIREMENT IS FOR EITHER:
VOL1 IN EBCIDIC FOLLOWED BY UP TO SIX EBCIDIC CHARACTERS.
OR SIX BYTES PRECEDING THE ABOVE THE FIRST FOUR OF WHICH ARE ZERO.
THIS COVERS IBM. AND AT LEAST THREE TYPES OF I.C.L. LABELS.

5) ROUTINE FREPORT BECOMES INTERNAL.

6) LABEL READING SPEEDED UP BY REDUCING FROM SIX STEPS TO FOUR

7) RELEASE/RETRIEVE CODE FOR V/MODE Max = ON/OFF.
AWAIT VOLUMES & GPL CHANGES FOR TESTING.
ANTICIPATE REDUCTION OF TAPE SIZE BY 1½K.

TAPE ENTRY FORMAT (AS OF TAPE 64A)

This format is private to the supervisor tape routine and is found mapped onto space allocated in the GPC communications segments.

INTEGER(0) LINK - Address of the next entry. (Zero = end of list)

BYTE (4) SNO - Device service number.

BYTE (5) USE - The bits of this field define the current use of the deck.

1<<5 = READ LABEL SEQUENCE
1<<4 = UNLOADING
1<<3 = I/O REQUEST OUTSTANDING
1<<2 = I/O REQUEST VALIDATED
1<<1 = REQUEST EXTENDED

ie. A user request is first checked (USE = 4), then issued to GPC (use = 12), and if error recovery is required further commands are sent (use = 14).

BYTE (6) LAST REPLY - A copy of the P_P2 flag sent to the user in reply to his last request.

BYTE (7) USER COUNT - Set to one if the deck has been claimed by an ENTRY 12.

INTEGER (8) EXERRS - EXERRS is proportional to the number of operations performed on the deck since allocation. If it is exceeded by the number of faults then a warning is given. -EXCESSIVE ERROR RATE.

INTEGER (12) FAULTS - The number of failures on this deck since its allocation.

STRING(3) (16) MNEM - Deck mnemonic eg M03

INTEGER (20) LBE1 - Logic block entries. Up to four
LBE2 tape commands chained together.
LBE3
(32) LBE4

BYTE (36) NEXT - Variable controlling label reading sequence.

0 = NOT IN PROCESS
1 = SEND PROPERTIES + SENSE
2 = REWIND
3 = READ BLOCK
4 = REWIND (IF UNLABELLED)

BYTE (37) PROP1 - Properties codes returned by
PROP2 SEND PROPERTIES command.
(39) PROP3

INTEGER (40)	LOAD	-	Status of deck -2 = INOPERABLE 0 = NOT LOADED 1 = LOADED WITH LABEL 2 = LOADED NO LABEL
INTEGER (44)	STREAMR0	-	Stream response words.
(48)	STREAMR1		These are the 'reply' from the hardware to the last command. (tok byte set = SNO)
INTEGER (52)	OPS	-	Number of READ/WRITE operations performed since allocation.
INTEGER (56)	ALE1	-	Address list entries.
	ALE2		These are two descriptors for
	ALE3		use in defining data for the
(68)	ALE4		LBE's.
BYTE (72)	SSTAT	-	Secondary status byte
BYTE (73)	T0	-	Tertiary status bytes.
	T1		
	T2		
	T3		
	T4		
	T5		
	T6		
	T7		
	T8		
	T9		
	T10		
(84)	T11		

RECORD REQ (88) . Details of current user request.

INTEGER (88)	DEST		
	SRCE		
	IDEN		
(100)	LENGTH	-	size of data
(104)	SPAN	-	address of data
BYTE (108)	CONTROL	-	Bits amend command (see SUP 17)
	TYPE	-	command
	DUM1		
	FAULTS	-	faults on this request
INTEGER (112)	LSL	-	Local segment limit
	LSTBA	-	Local segment base address
	COUNT1	-	Used for counting blocks & TM's.
(124)	COUNT2		"

RECORD RCB (128) - Request control block

INTEGER (128)	LIMFLAGS	-	see GPC spec.
	LSTBA		
	LBBYTES		
	LBA		
	ALBYTES		
	ALA		
(152)	INITWORD		
INTEGER (156)	LAB1	-	Space used for label reads.
	LAB2		
STRING (7) (164)	DSN	-	Label

BYTE (172) ORGDIR - Original direction of movement.
1 = FORWARD
0 = BACKWARD

BYTE (173) UNLOADPENDING - IF SET then unload deck when current command terminates.

BYTE (174) ORGREQ - Original command
Note/ REQ_TYPE is overwritten during error retry etc.

BYTE (175) BITS Defines mode and B.P.I.

INTEGER (176) FIRSTPEND - Start of queue of pending requests.
LASTPEND End " " "

Further information on several of these fields can be found in the GPC descriptions and in Supervisor note 17.

Alan Anderson 21/4/80

Supervisor (TAPES) Release Note.

Version - 75

Date - 17 March 82

Source - ERCC15.TAPE75S

This release is a complete rewrite of the tape handler, and the opportunity has been taken to remove some historical eccentricities from the specification. I believe that no existing software will be affected by these changes, but anyone who thinks differently should contact me soon. The differences are detailed in section 4.

P or S series hardware is supported, depending on compile time option SSERIES.

Contents:-

Section 1 - Activities
Section 2 - Queueing of requests
Section 3 - Private chains
Section 4 - Differences from previous specification

1. Activities

Claim tape - activity 12 (decimal)

This activity will generally be used directly only by Chopsupe or Volumes either on its own or a user's behalf.

Call:-

format - integer DEST,SRCE,IDENT,PERM,string(7)DSN,string(3)MNEM, c
byteinteger D1,D2,D3,MODE

P_DSN - the required tape is identified by P_DSN unless this is "NOLABEL"
P_MNEM when the tape is that loaded on device P_MNEM. The latter is not otherwise required.

P_PERM - read/write access is specified by P_PERM=1/2 respectively.

P_MODE - this is in two parts; P_MODE&3 specifies the override mode, while P_MODE&4 specifies use of the compress expand facility. The current mode of a tape is the recording mode, PE or NRZI, in which the tape was last recorded. This is detectable by the hardware while the tape is at BT. In the label reading sequence performed whenever a tape is loaded, the current mode is established as PE, NRZI or undefined (the latter will typically be for a blank tape). If P_MODE&3 is zero, the current mode so established will be used in all accesses to the tape. If P_MODE&3 is 2(NRZI) or 3(PE), this is used instead of the current mode; however any request other than a write at BT with a mode different from the current mode will generate a hardware failure. It is not therefore possible for a tape to be recorded in different modes at different positions along its length, since it is not possible to move away from BT other than by writing, which changes the current mode to the new mode. Typically, the override mode will only need to be specified when the current mode of a tape is to be changed by a write at BT, typically a relabelling operation. Volumes provides facilities to label a tape in either mode; the tape will then be accessed automatically in that mode.
The compress expand facility allows four 6-bit bytes to be recorded as three 8-bit bytes; this is specified by P_MODE&4 which is ORed into the override or current mode as appropriate. It is possible for some files on a tape to be recorded in CE mode while others are not, though only one kind can be successfully accessed in any one 'claim'.
P_MODE is specified by the user in the second byte of the mode given to 'DMAG CLAIM', the first byte being READ/WRITE. Existing software in which this byte is zero, will cause the tape to be accessed in the current mode without CE, which will be correct in most cases.

Reply:

format - integer DEST,SRCE,IDENT,FLAG,SNO,byteinteger D1,D2,D3,DEVID, c
integer P5,P6

P_FLAG - 0=success, when
P_SNO is the service number to be used in subsequent requests, and
P_DEVID is the mechanism address which can be used for advisory purposes to identify the deck.
1=tape(or mechanism) not found or already in use.
2=no write ring fitted and write access requested.
3=mode error; invalid override mode, or NRZI override mode specified and not fitted, or no override mode specified and current mode is undefined.

Execute request - activity as in SNO returned in reply to claim.

Call:

format - integer DEST,SRCE,IDENT,halfinteger LENGTH, c
byteinteger CONTROL,TYPE,integer ADDRESS,AUXLENGTH,LSL,LSTBA

P_TYPE - the type of request:
0 - erase
1 - read one block forward
2 - write one block
4 - read-check one block forward
5 - private chain
6 - read one block reverse
8 - skip n blocks forward or reverse
9 - skip n tape marks forward or reverse
10 - write tape mark
17 - rewind

The interpretation of the remaining fields is type dependent. They are summarised here and detailed later.

P_LENGTH - a signed count of bytes, blocks or tape marks; if zero P_AUXLENGTH is used instead, to accommodate counts >=32K.

P_CONTROL - a byte either
whose individual bits are interpreted where appropriate as follows:
2**0 - suppress tape mark checking
2**1 - suppress error retry
2**2 - suppress short block checking
2**3 - suppress long block checking
or
whose value specifies the number of tape marks/logical file.

P_ADDRESS - the virtual address of the area for a data transfer.

P_LSL
P_LSTBA - local segment limit and table base defining a locked VM containing the data area for a transfer; typically supplied by a user's local controller on an OUT18; but see section on private chains.

Reply:

format - integer DEST,SRCE,IDENT,FLAG,LENGTH,AUXLFAG, c
byteinteger T0,T1,T2,T3,T4,T5,T6,T7

The interpretation of the fields is dependent on the type of the original request. They are summarised here and detailed below.

P_FLAG - 0=request successful; qualified by P_AUXFLAG
1=request failed(short,long or media error), but it is safe to continue; qualified by P_AUXFLAG. This will only occur on a transfer request.
2=request rejected without being executed; qualified by P_AUXFLAG. This can occur on any request.
3=request failed and it is unsafe to continue as position may have been lost. This can occur on any request.
4=request successful, except that a tape mark, double tape mark, BT or ET was detected. This can occur on a transfer, skip block or skip tape mark request.

P_LENGTH - a count of bytes transferred, or blocks or tape marks skipped; always >=0 except for a read reverse short (see below).

P_AUXFLAG - if P_FLAG=0, either 1=short or 2=long
if P_FLAG=1, either 1=short, 2=long or 4=CE error
if P_FLAG=2, either 1=bad params, 2=tape not (no longer) accessible or, 4=SNO not found; the latter two will typically be the result of a deck going unavailable then possibly being reallocated to another user.

P_T0-T7 - tertiary status bytes as at latest abnormal termination.

Requests:

ERASE (type=0)
A length of tape equivalent to P_LENGTH bytes is erased.

READ FORWARD (type=1)
One block is read forward, up to P_LENGTH bytes being transferred to store ascending from P_ADDRESS; if the tape block is shorter or longer than P_LENGTH, P_AUXFLAG will be returned=1(short) or 2(long), and, unless the relevant bits in P_CONTROL were set in the request to suppress this condition, P_FLAG will be returned=1; P_LENGTH will be returned=the actual number of bytes transferred; the actual length of a long block is not available.
In the event of a recoverable error(typically parity), a retry sequence involving up to 15 retries in both directions will be initiated unless the relevant bit in P_CONTROL is set to suppress it; in reverse, up to P_LENGTH bytes are transferred to store descending from P_ADDRESS+P_LENGTH; if a read succeeds in reverse and also the block is short, the data will be at the 'top' of the data area, and to signal this, P_LENGTH is returned=-the actual number of bytes transferred; if the error persists to the end of the retry sequence, P_FLAG will be returned=1 and P_AUXFLAG will be returned with neither short nor long; although some or all of the data may have been transferred, P_LENGTH is returned=0.
In the specific case of a bad CE data count(CE transfers must be in multiples of 4), P_FLAG will be returned=1, P_AUXFLAG will be returned=4 and no retry sequence is initiated.
In the event of an irrecoverable error, P_FLAG is returned=3; continuation is at the user's risk as position may be lost.
If a tape mark is detected instead of a data block, P_FLAG is returned=4 after repositioning the tape before the tape mark.

WRITE (type=2)
One block is written to the tape, P_LENGTH bytes being transferred from store starting at P_ADDRESS.
In the event of a recoverable error, unless the relevant bit in P_CONTROL is set to suppress it, a retry sequence is initiated consisting of a skip reverse over the badly written block, erasure of its length and a rewrite; this is repeated up to 6 times, the rewrite being attempted progressively further up the tape; if the error persists, P_FLAG is returned=1.
In the case of a bad CE data count, P_FLAG is returned=1, P_AUXFLAG is returned=4 and no retry sequence is initiated.
In the event of an irrecoverable error, P_FLAG is returned=3; position may be lost so continuation is at the user's risk.
If the ETW marker is detected during the write or retry sequence, but the write is otherwise successfull, P_FLAG is returned=4.

READ CHECK (type=4)

One block is read forward; only P_CONTROL is relevant, and only the 'suppress error retry' bit; a read length of 256K-1(P series) or 64K-1(S series) is generated with data transfer and short block checking both suppressed; this proceeds as for read forward, except that if the request is successfull, P_AUXFLAG is returned=the number of retries required.

READ REVERSE (type=6)

One block is read in reverse, up to P_LENGTH bytes being transferred to store descending from P_ADDRESS+P_LENGTH; this proceeds as read forward except that if BT is detected instead of a data block, P_FLAG is returned=4 after repositioning.

SKIP n BLOCKS FORWARD or REVERSE (type=8)

P_LENGTH may not be zero. If P_LENGTH>0, that many blocks are skipped forward otherwise they are skipped reverse.
If a tape mark is detected, P_FLAG is returned=4 after repositioning the tape to the correct side of the tape mark (ie the position as it was before the tape mark was traversed), unless the 'suppress tape mark checking' bit is set in P_CONTROL, when the tape mark counts as a block; additionally in reverse, if BT is detected P_FLAG is returned=4 after repositioning.
P_LENGTH is returned=the actual number(unsigned) of blocks (and tape marks) skipped.
In the event of any other error, P_FLAG is returned=3 and continuation is at the user's risk as position may be lost.
A special condition can arise when skipping forward with 'suppress tape mark checking' set in P_CONTROL. No look-ahead is performed for adjacent tape marks (see below), and it is thus possible to pass the logical end-of-tape thus signalled.

SKIP n TAPE MARKS FORWARD or REVERSE (type=9)

P_CONTROL is the number of tape marks/logical file and may not be zero;
P_LENGTH is the number of logical files to be skipped and may not be zero;
mod(P_LENGTH)*P_CONTROL tape marks are skipped, forward if P_LENGTH>0 otherwise reverse.
For a forward skip, if a tape mark is the first thing detected, P_FLAG is returned=4 after repositioning the tape before the tape mark; if subsequently two adjacent tape marks are detected, P_FLAG is returned=4 after positioning the tape between them; it is not therefore possible to pass the logical end-of-tape signalled by two adjacent tape marks.
If BT is detected when skipping reverse, P_FLAG is returned=4.
P_LENGTH is returned=the actual number(unsigned) of tape marks skipped.
In the event of any other error, P_FLAG is returned=3 and continuation is at the user's risk as position may be lost.

WRITE TAPE MARK (type=10)

One tape mark is written to the tape; this proceeds as for write, except that the setting of 'suppress error retry' in P_CONTROL is ignored.

REWIND (type=17)

The tape is rewound to BT.

Release tape - activity 7

This activity will generally be used directly only by Chopsupe and Volumes.

Call:-
  format - integer DEST,SRCE,P1,halfinteger H1,SNO,integer P3,P4,RETAIN,P6

  P_SNO   - the service number of the tape to be released, as returned in the
             matching 'claim'.

  P_RETAIN - the tape will be unloaded/rewound to BT and the label reading
             sequence reinitiated, if P_RETAIN=0/1 respectively.

Reply:-
  format - integer DEST,SRCE,P1,FLAG,P3,AUXFLAG,P5,P6

  P_FLAG   - 0=release successfull
              2=release unsuccessfull, qualified by P_AUXFLAG

  P_AUXFLAG - 2=tape not claimed
              4=SNO not found

Notify decks on/off - activity 4

This activity will be used once /cluster by GPC(DCU) at startup, and
thereafter by Volumes in response to the operator MODE command.

Call:-
  format - integer DEST,SRCE,P1,P2,P3,P4,P5,P6

  P_P1 - base mnemonic
  P_P2 - 0/1=off/on respectively; only relevant for Volumes calls; GPC
         calls are assumed to mean 'on'.
  P_P3 - number of decks upwards from base mnemonic; relevant only for Volumes
         calls; for GPC calls, the maximum decks/cluster is assumed.

  One allocate/deallocate request per deck is sent to GPC, and on receipt
  of GPC's reply, if mode=on, the deck table area is initalised and linked in
  and a load sequence initiated to get property codes and determine the deck
  status, or if mode=off, the deck table area is unlinked.

Reply:-
  A reply is sent only if the caller was Volumes, when P_P1 is the number of
  decks NOT acted on; ie those found(already 'on') if request was 'on', or
  those not found(already 'off') if request was 'off'.

Receive Volumes SRCE - activity 6

This activity is used once by Volumes at startup to report present and ready
to receive load messages.

  P_SRCE - the service number to which load messages are to be sent.

Display decks status - activity 13

This activity is used by Volumes in response to an operator TAPES command.
The states of all known decks is displayed.

Set reporting level - activity 8

If P_P1<=7 it sets the reporting level as follows:
  2**0 - report parm records in
  2**1 - report parm records out
  2**2 - dump deck table whenever a chain is fired
If P_P1=16/17 then set read/write retries(default 15/6) to P_P2&X'FF'. Note that these apply to all decks.

Send failure details - activity 9

Who uses this activity?

Stream response and status bytes are returned for the deck identified by the SNO in P_P1&X'FFFF'.
If P_P1 is returned=0, then
     P2 is stream response 0
     P3 is stream response 1
     P4,5,6 are tertiary status bytes T0-T11.
If P_P1 is returned=2, the SNO was not found.

Dump deck table - activity 10

The tabel for the deck identified by the SNO in P_P1&X'FFFF' is dumped to the mainlog.

2. Queuing of requests

Execute requests received while a deck is busy are queued until either their turn for execution comes, or a state change renders them unexecutable. Replies are sent normally and care must be exercised in the interpretation of failures, since failure of one request does not inhibit execution of any queued.

3. Private chains

These are currently implemented only for P series.

Once a tape has been claimed in the normal way, the user may issue Execute private chain requests. These may be freely interleaved with normal requests. In a private chain request, the user assumes entire responsibility for setting up the RCB, LBEs, ALEs and data areas. The RCB is merely copied to the device area and the request fired. On termination, the stream response and status bytes as received from GPC are passed back to the user, who is responsible for their interpretation. Optionally the user may also undertake to deal with attentions from the deck; this only applies until the next request is received and only applies to BT and tape mark attentions; if the deck status changes, control reverts to the supervisor (including the case where a private chain unloads the deck).
All the data areas - RCB, LBEs, ALEs and buffer areas must be in a locked VM created by a call of DLOCK, and the segment limit and table base returned by DLOCK must be supplied unaltered with the request.

```
Call:-
  format - integer DEST,SRCE,IDENT,byteinteger D1,D2,D3,TYPE, c
           integer RCBADDR,ATTNDEST,SL,STB

  P_TYPE     - 5
  P_RCBADDR  - virtual address of RCB
  P_ATTNDEST - if #0 the destination to which BT and tape marks attentions
               are forwarded
  P_SL       - segment limit and table base as returned from DLOCK for RCB
  P_STB

Reply:-
  format - integer DEST,SRCE,IDENT,FLAG,P3,P4, c
           byteinteger T0,T1,T2,T3,T4,T5,T6,T7

  P_FLAG - 2=request rejected without execution, when P_P4=1,2 or 4 for
             bad params, tape not accessible or SNO not found respectively
  P_FLAG - 0,3=request executed, when
  P_P3   - stream response 0
  P_P4   - stream response 1
  P_T0-7 - tertiary status bytes

Attentions are forwarded exactly as received from GPC; ie
  P_P1   - stream response 0, the attention byte at P1>>8
  P_P2   - stream response 1.
```

4. Differences from former specification

a) The mode specification in 'claim tape' is additional; however a zero value will cause the tape to be accessed in the current mode without compress/expand.

b) A negative length in an execute read forward request is no longer accepted; it was formerly interpreted as a read reverse.

c) Bit 2**7 in the control byte of an execute request formerly indicated PE/NRZI mode; this bit is now ignored, given the more comprehensive mode facilities in 'claim tape'.

d) 'Release tape' identifies the tape by the SNO only; formerly it could be identified either by the SNO or the DSN.

e) In activity 9, 'send failure details', the deck is identified by the SNO in P_P1&X'FFFF'; it was formerly identified by the SNO in P_P1>>24.

f) Activity 10 is now 'dump tape table' as described above; it was formerly 'set load state' which is not now implemented.

g) Activity 11 is no longer implemented; it was formerly 'sense and read label' on deck identified in P_P1>>24.

..... C.McArthur 23 March 1982

Table of Service numbers for MAXPROCS = 128

Proc	SYNC1	SYNC2	ASYNC	Proc	SYNC1	SYNC2	ASYNC
1	X0C1	X141	X1C1	65	X101	X181	X201
2	X0C2	X142	X1C2	66	X102	X182	X202
3	X0C3	X143	X1C3	67	X103	X183	X203
4	X0C4	X144	X1C4	68	X104	X184	X204
5	X0C5	X145	X1C5	69	X105	X185	X205
6	X0C6	X146	X1C6	70	X106	X186	X206
7	X0C7	X147	X1C7	71	X107	X187	X207
8	X0C8	X148	X1C8	72	X108	X188	X208
9	X0C9	X149	X1C9	73	X109	X189	X209
10	X0CA	X14A	X1CA	74	X10A	X18A	X20A
11	X0CB	X14B	X1CB	75	X10B	X18B	X20B
12	X0CC	X14C	X1CC	76	X10C	X18C	X20C
13	X0CD	X14D	X1CD	77	X10D	X18D	X20D
14	X0CE	X14E	X1CE	78	X10E	X18E	X20E
15	X0CF	X14F	X1CF	79	X10F	X18F	X20F
16	X0D0	X150	X1D0	80	X110	X190	X210
17	X0D1	X151	X1D1	81	X111	X191	X211
18	X0D2	X152	X1D2	82	X112	X192	X212
19	X0D3	X153	X1D3	83	X113	X193	X213
20	X0D4	X154	X1D4	84	X114	X194	X214
21	X0D5	X155	X1D5	85	X115	X195	X215
22	X0D6	X156	X1D6	86	X116	X196	X216
23	X0D7	X157	X1D7	87	X117	X197	X217
24	X0D8	X158	X1D8	88	X118	X198	X218
25	X0D9	X159	X1D9	89	X119	X199	X219
26	X0DA	X15A	X1DA	90	X11A	X19A	X21A
27	X0DB	X15B	X1DB	91	X11B	X19B	X21B
28	X0DC	X15C	X1DC	92	X11C	X19C	X21C
29	X0DD	X15D	X1DD	93	X11D	X19D	X21D
30	X0DE	X15E	X1DE	94	X11E	X19E	X21E
31	X0DF	X15F	X1DF	95	X11F	X19F	X21F
32	X0E0	X160	X1E0	96	X120	X1A0	X220
33	X0E1	X161	X1E1	97	X121	X1A1	X221
34	X0E2	X162	X1E2	98	X122	X1A2	X222
35	X0E3	X163	X1E3	99	X123	X1A3	X223
36	X0E4	X164	X1E4	100	X124	X1A4	X224
37	X0E5	X165	X1E5	101	X125	X1A5	X225
38	X0E6	X166	X1E6	102	X126	X1A6	X226
39	X0E7	X167	X1E7	103	X127	X1A7	X227
40	X0E8	X168	X1E8	104	X128	X1A8	X228
41	X0E9	X169	X1E9	105	X129	X1A9	X229
42	X0EA	X16A	X1EA	106	X12A	X1AA	X22A
43	X0EB	X16B	X1EB	107	X12B	X1AB	X22B
44	X0EC	X16C	X1EC	108	X12C	X1AC	X22C
45	X0ED	X16D	X1ED	109	X12D	X1AD	X22D
46	X0EE	X16E	X1EE	110	X12E	X1AE	X22E
47	X0EF	X16F	X1EF	111	X12F	X1AF	X22F
48	X0F0	X170	X1F0	112	X130	X1B0	X230
49	X0F1	X171	X1F1	113	X131	X1B1	X231
50	X0F2	X172	X1F2	114	X132	X1B2	X232
51	X0F3	X173	X1F3	115	X133	X1B3	X233
52	X0F4	X174	X1F4	116	X134	X1B4	X234
53	X0F5	X175	X1F5	117	X135	X1B5	X235
54	X0F6	X176	X1F6	118	X136	X1B6	X236
55	X0F7	X177	X1F7	119	X137	X1B7	X237
56	X0F8	X178	X1F8	120	X138	X1B8	X238
57	X0F9	X179	X1F9	121	X139	X1B9	X239
58	X0FA	X17A	X1FA	122	X13A	X1BA	X23A
59	X0FB	X17B	X1FB	123	X13B	X1BB	X23B
60	X0FC	X17C	X1FC	124	X13C	X1BC	X23C
61	X0FD	X17D	X1FD	125	X13D	X1BD	X23D
62	X0FE	X17E	X1FE	126	X13E	X1BE	X23E
63	X0FF	X17F	X1FF	127	X13F	X1BF	X23F
64	X100	X180	X200	128	X140	X1C0	X240

Roderick McLeod

System

EMAS 2900 SYSTEM NOTE

No: 1
Date: 6/5/77

Fixed Sites on EMAS System Discs

EPAGE NUMBER	LENGTH (EPAGES)	USE
0	X40	IPL SUPERVISOR (CHOPSUPE)
X40	X40	SUP VSN 0
X80	X40	1
XC0	X40	2
X100	X40	3
X140	X40	4
X180	X40	5
X1C0	X40	6
X200	X40	DIRECTOR VSN 0 (default)
X240	X40	1 (DIRVSN 1)
X280	X40	2
X2C0	X40	3
X300	X100	Default BASEFILE (or room to get 2 versions if required and neither exceeds 512K). IMP Compiler? No, just enough to RELOAD into the file system. (Process 1 DIRECTOR to look after the file system initialisation and maintenance)
X400	X100	Local controller stack, DIRECTOR stack, and gla for Process 1 (sub-sites to be assigned by Dave).

File system to start at X800 (or lower). (Currently it starts at X2000).

J.K. Yarwood

EMAS 2900 SYSTEM NOTE

No: 2 (revised)
Date: 10/5/79

Configuration and Grope

EMAS 2900 attempts to define the configuration on which it is running by the "grope" feature and to adapt (with limits) accordingly. This Note defines the current limits and the grope procedure.

The grope procedure has the great advantage of allowing one Supervisor to be used for all sites, thus minimising maintenance. It has the minor operational disadvantage that devices powered down or physically disconnected are missed during the grope and cannot be used until the next IPL. Devices in manual or marked not available are discovered and incorporated into the configuration.

Configurations

* OCP and SACs

  EMAS 2900 is currently restricted to one OCP and one SAC. It is simple to extend the software to multiple SACs but there is a real difficulty in the smaller machines where the SAC selection logic is omitted and the same SAC appears on all vacant ports. Since a single SAC has so much more usable transfer capacity than a 4-75, extensions to multiple SACs have a very low priority.

* Core and SMACs

  EMAS 2900 can accept these without limit. There must be at least 0.5 Megabyte and there must be at least 256K on SMAC 0. The advantages of having more than the minimum number of SMACs for any given core size are unquantifiable but probably marginal. The amount of core require depends on the OCP speed - probably 1 Mb on 2960, 1.5 Mb on 2970 and 2 Mb on 2976 or 2980 is a reasonable minimum.

* SFCs or Drums

  EMAS 2900 can run without a paging device but this would not give satisfactory multi-access except on a 2960 and smaller machines. For larger machines a ratio of between 5 and 10 Mb of paging storage per megabyte of core store will give a balanced configuration - the lower figure when substantial batch capacity is envisaged, the larger for 100% MAC machines.

* Discs

  EMAS 2900 supports only the larger discs, EDS100 and EDS200, with each disc being treated as a logical file system; EDS200s are preferable to EDS100s. EDS30s and 60s are too small. The number of drives required depends on the file space to be allowed to each user and on the number of accredited users. EMAS 4-75 allows each user over 10 Mb but the average is a little over 1 Mb. This figure may reflect the Edinburgh teaching load, and other installations will probably wish to allow 2-5 Mb/user. The accredited users should be spread as evenly as possible over the

available logical file systems so that each file system has its share of serious and casual users.

* GPCs

EMAS 2900 will support any number of GPCs, although the advantages of having more than one seem marginal. Each EMAS 2900 configuration must have at least one OPER (and preferably a two-screen OPER) and one tape cluster of two decks. Larger configurations will require two to four clusters for back-up and archiving. All other slow peripherals are best provided on the Front End Processor, although one local card reader and one local line printer (for diagnostics) are supported.

* FEP

A Front End Processor is required to handle all terminal and RJE traffic.

Groping

This process takes place in several steps and takes about 15 seconds as there have to be a lot of software timeouts. Before starting to grope it is essential to ensure that:

a) time-out is enabled on the Engineer's panel.

b) any line printers have the right cartridge no. set (see Step 3).

After this an IPL load will load CHOPSUPE, which starts the grope steps.

Step 1

CHOPSUPE tries to access all 16 possible SMACs - non-existent SMACs time out. Any SMACs present are then asked to send their configuration registers. From these CHOPSUPE builds the core tables and clears all core (except the first 256K of SMAC 0).

Step 2

CHOPSUPE determines the port(s) containing SACs. All trunks on the SAC(s) are then put into direct control mode and the controllers told to identify themselves. A list is made of the controllers relevant to EMAS 2900 - the others are ignored.

Step 3

All GPCs are initialised and their microprograms are loaded. Property codes are read for all streams. All OPERs are cleared and the the OPER with the largest number of screens is selected as the principal OPER. Reps are loaded into all printers according to the following table:

Cartridge no. (set on handswitch)	Rep
0	The currently loaded rep, if any; otherwise the 64 character rep
2	The standard 48 character rep
3	The non-standard 384 character rep for the Bush 2980
4	The standard 64 character rep
5	The standard 96 character rep

The first printer found is selected as the diagnostic printer. Any operable tapes are rewound and the labels read.

Step 4

All SFCs are initialised and their microprograms loaded. Any operable devices are formatted and the available number of page frames calculated.

Step 5

All DFCs are initialised and the devices examined. Any EDS100s and EDS200s are entered into a table and a label-reading sequence initiated. EDS30s and 60s are ignored.

Step 6

A communication area (Public Segment 48 - further details in EMAS 2900 Supervisor Note 15) is constructed with the information gleaned ready for handing to the Supervisor. A summary of the configuration is displayed on the principal OPER and the first Operator Command awaited.

P.D. Stephens

OPER

Each 2900 OPER station comprises a keyboard and one or more screens, up to a maximum of four. Each screen can display twenty-four lines (numbered from 0 at the top to 23 at the bottom), each containing forty characters (numbered from 0 on the left). One screen is connected to the keyboard and is referred to as the "right-hand screen". The hardware echoes the characters typed at the console on line 22 of this screen.

To simplify the software, EMAS 2900 regards each OPER station as a number of virtual displays. This number may vary - currently it is six - but it is independent of the number of actual screens on the OPER station.

The six virtual displays are as follows:

0 Last 21 messages from the System (scrolled) plus interactive facilities for entering Commands from keyboard.

1 Status of 1st 60 processes plus System queue information.

2 Status of 2nd 68 processes.

3 Reserved for VOLUMS and DIRECT.

4 Reserved for SPOOLR (see separate documentation).

5 The 24 messages from the System immediately previous to those on screen 0.

For normal multi-screen OPERs the right hand screen is locked onto virtual display 0 and the left hand screen starts with virtual display 1. Use of the "page forward" and "page back" keys on the OPER keyboard increases or decreases the number of the display on the left hand screen; the list is treated as cyclic - going past the last virtual display switches back to the first.

For single screen OPERs the only screen starts on virtual display 0. The page forward and back keys map all six virtual displays onto the screen. An additional feature is that pressing the "ENTER" key causes an immediate switch to virtual display 0, as does the arrival of a message or prompt for the interactive screen (i.e. virtual display 0).

The twenty-four lines of the interactive screen are used as follows. The "activities" referred to are explained below, in the description of externalroutine OPER.

lines 0-20 - scrolled output (from activity 7 on OPER)

line 21 - prompt line (from activity 8)

line 22 - input line (hardware)

line 23 - input echo (software)

Two routines are provided for interfacing:

1) DISPLAY TEXT for updating virtual screens 1-6, and

2) OPER for interactive facilities (i.e. virtual screen 0)

```
1) externalroutine DISPLAY TEXT (integer WHICH, LINE, CHAR, c
                                 string (41) TEXT)
```

The text on line LINE of OPER number WHICH starting from character position CHAR is set to TEXT.

```
 Lines  0-23 are on virtual screen 1
       24-47 are on virtual screen 2
       48-71 are on virtual screen 3
         .
         .
```

The third and fourth screens, when present, display virtual displays n+1 and n+2 respectively, where virtual display n is on screen 2. The addition is cyclic as for "page forward".

```
2) externalroutine OPER(recordname P)

   recordformat PFM(integer DEST, SRCE, c
                    integer P1,P2,P3,P4,P5,P6
                    string(23) TEXT
                    byteinteger LINE,CLEAR,ZERO, string(20) DTEXT

P_DEST = X'32'<<16 + operno<<4 + activity
```

operno: (0 <= operno <= 15) specifies which OPER is required; zero selects the principal OPER

activity:

1 - set monitoring level from P_P1; 0 (no monitoring) is the default

2 - normal termination (from GPC)

3 - attention (from GPC)

4 - initialize (from GPC)

5 - abnormal termination (from GPC)

6 - call DISPLAY TEXT as follows:

DISPLAY TEXT(WHICH, P_LINE, P_CHAR, P_DTEXT)

If P_ZERO=255, first clear the whole of the relevant virtual screen.

If P_ZERO=1, operate "page forward" by software until the virtual screen containing P_LINE is visible.

7 - output request, P_TEXT = message. The message is scrolled onto the right hand screen, preceded by "SRCE/" with SRCE in Hex.

8 - input request, P_TEXT = prompt string. The input is ponned back to "SRCE" in P_TEXT (two messages will be sent if the input line is longer than 23 characters).

9 - update left hand screen (kicked by the clock).

10 - nominate main OPER
P_P1 = new main OPER "operno". The specified OPER becomes the principal OPER. P_P1 = -1 makes SRCE (i.e. this) the current OPER.

Operational notes

1) The OPER accepts input in two modes:

   * following an input request (activity 8)

   * after "COMMAND" has been struck

   In both cases the appropriate prompt is displayed and the input line may be typed (and terminated by "ENTER").

   Requested input is sent to the service which made the request.

   Command input is sent to the resident command interpreter (routine PARSE COM).

2) Striking "COMMAND" will force the OPER into command input mode. Any input request which is pre-empted will be re-established once the command line has been input.

3) A command input line may have continuations. An ampersand (&) at the end of a line is the continuation mark.

P.S. Robertson
P.D. Stephens

EMAS 2900 System Calls

EMAS supervisors from Sup. 11F onwards have support code for the system call mechanism. The larger 2900s have a hardware mechanism for inward calls but not outward calls. The software support for inward calls has two purposes:

1) To appear identical to the hardware mechanism in the case of the smaller machines and also emulating machines where pressure on the micro-program store has led to the omission of the hardware mechanism.

2) To provide more information if the hardware call fails.

Inward calls are completely defined by section 5.3.15 of the PLI SD 2.5.1) and are not considered further in this note.

Outward calls are to enable the Subsystem to run user programs at a lower level of privilege and on a separate stack. Software intervention is needed to copy any parameters and to allow for an inward return - since the EXIT instruction cannot lower the ACR level.

Before making an outward call the Subsystem must

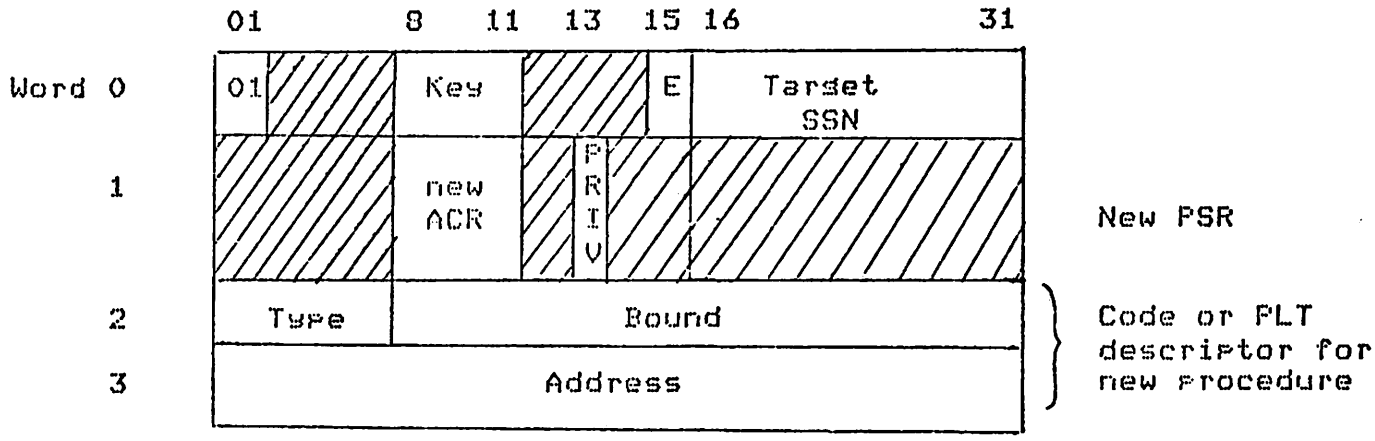
1) Create the new (Target) stack.

2) Notify the Local Controller of the stack so that it can provide the locked-down "SSN+1" segment.

3) Ensure that its current stack will not be accessible to the user at his higher ACR level.

4) Make a suitable entry in the system call tables.

The system call table entry for an outward call is as follows:



The Key is checked against the caller's ACR. If satisfactory, the procedure defined by the descriptor is entered on the new stack with B, C and the parameters set up on the old stack. A special inward return descriptor is set in the LINK field of the new stack which does the converse. ACC and B are preserved across an inward return so that it is possible to outward call Function and Maps if desired.

Bit 15 of word 0 of the system call table entry distinguishes ICL and EMAS conventions. If this bit is 0 the local name space of the called procedure starts at word 0 of the new stack (ICL convention). If Bit 15 is not 0 then the local name space starts after the "useful" information in the file, as defined by word 0 of an EMAS standard header. This permits "preloaded" Fortran stacks if these are going to be supported.

Error Codes

All system call failures give a system call error (16). The subclass signifies as follows:

^ = "I" Value out of range.

↓ = "J" value out of range.

2 = ACR check fails - user insufficiently privileged to use this system call.

3 = New stack invalid - validate fails on stack, or SSN+1 not resident, or new stack = current stack.

4 = Task calls not implemented.

5 = Inward return from procedures not activated by an outward call.

33 = ACR fails: outward call new ACR < current ACR, or inward call new ACR > current ACR.

34 = Invalid Code/PLT descriptor in SCT entry.

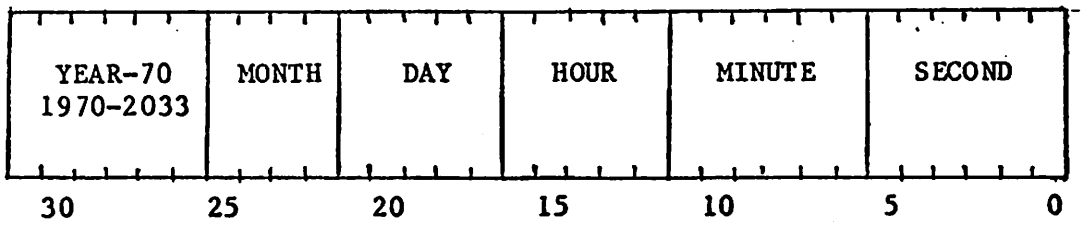
P.D. Stephens

No: 5
Date: 17/4/78

Proposal for a standard format for packed date and time

There are a number of places in the Subsystem, Director, and System Processes where it is useful to store the date or the date and time, in a packed format. I suggest that the following format, although inelegant, has the merits of being decoded with a trivial function, keeps both date and time in 32 bits, and can be used for sorting into chronological order. For dates beyond 2001 it will be necessary to sort on all 32 bits, treating the most significant bit as 2**31.

The format is:



When used for Date alone the 17 least significant bits should be all 0. The file

ERCC06.NR_PACKDTS

contains the source of the following functions, which operate with respect to the format.

stringfn UNPACKDATE(integer PACKED)

stringfn UNPACKTIME(integer PACKED)

These return strings with the standard formats using "/" as separators in the date and "." as separators in the time.

integerfn PACKDATE(string (8) DATE)

integerfn PACKDATEANDTIME(string (8) DATE,TIME)

Roderick McLeod

EMAS 2900 SYSTEM NOTE

No: 6
Date: 24/1/80

Bootstrapping the EMAS 2900 System

These instructions apply to the P2, P3, P4 and P4/1 processors. Where a processor-specific operation is required in the load sequence the processor type or types for which the action is appropriate is indicated in brackets.

1. By inspection at the Engineer's panel (P3, P4, P4/1) or Engineer's control processor (P2), ensure that:

   a) timouts are enabled

   b) the handkey switches for image store data are set to zero

   c) Stop on SMAC failure is set (P3)

   d) μprog load is inhibited (P4)

   e) all other switches are in the central position (P3, P4, P4/1)

2. Configure the store as required, but note that:

   a) there must be at least 256K of contiguous store on SMAC0 starting at real address 0 to allow CHOPSUPE (the Supervisor loader) to load

   b) apart from requirement (a) the store need not be contiguous

   c) the store may be configured as being interleaved (P4, P4/1)

3. Ensure that any unused SAC ports are <u>not</u> connected back-to-back.

4. The printer repertoire to be loaded is determined from the cartridge number set on the printer thumbwheel, viz:

   0 - leave the currently loaded rep loaded, or load the 64 character rep if no rep is loaded (this rep comprises the first 64 characters of the 96 character rep)
   2 - load the 48 character rep for the PODPS Craiglockhart 2970
   3 - load the 384 character rep for the Bush Estate 2980
   4 - load the standard 64 character rep
   5 - load the standard 96 character rep (as for the ERCC KB 2970)

5. If it is not already loaded, load the standard OCP microprogram.

6. Mount the System IPL tape and perform the standard IPL sequence (P3, P4, P4/1) or autoload (P2) sequence <u>twice</u> - the first time to skip over the tape label, and the second time to load and enter the bootstrap program.

7. The bootstrap reads CHOPSUPE from the tape, rewinds the tape to BT and enters CHOPSUPE. The configuration is groped and after a short delay (-10 seconds) configuration details are displayed on the control Oper and the main line printer.

General note: The remaining instructions entail the use of the Oper console. Press COMMAND before typing a line, and press ENTER to enter it after it has been typed.

8. The EDS100 selected to contain the file system should now be formatted, thus:

```
FORMAT EDnn -1 -1
```

This will format the whole of the disc, indicating on the Oper if there are any write failures. These failures should be noted for use subsequently in section 13. Provided that there are no failures below cylinder number 19, continue with the system set-up; otherwise another disc will have to be used.

When the format is complete (after -8 minutes) an IPL label should be written to the disc, thus:

```
ILABEL EDnn <label>
```

See Director Note 4 for label conventions.

9. Now prime down the System from the IPL tape to disc:

```
PRIME <tape> <disc> X400
```

10. Inform the System of the current date and time:

```
DT ddmmyy hhmm
```

This may not be necessary on machines where the clock is preserved through IPLS (P4 now; others in future).

11. Director version 1 should be selected:

```
DIRVSN 1      (see Director Note 9)
```

This prevents an automatic consistency check being applied to the file system by the DIRECT process when the Supervisor is loaded, nor will the VOLUMS and SPOOLR processes be started.

12. The main Supervisor can now be loaded:

```
SLOAD <fsys> <site>
```

where fsys is the file system number - that is, the numeric part of the disc label; e.g. EMAS00 would be FSYS 0.

site is the start page number of the required Supervisor; e.g. X80.

13. The Supervisor creates the DIRECT process. A new file system can now be created by using the following Director commands (see Director Note 10). In each case n is the decimal FSYS number.

    a) D/CLEAR FSYS n

       The responses are: 1/BAD PGS CT=0
                          1/VOL/SPLR IND CRE 0 SUCC
                          1/DONE

    b) If there were any bad tracks discovered during formatting, enter them in the bad pages list:

       D/BADFSYSCYLTRK <fsys> <cyl> <track>

       The response is 1/DONE.

14. Next perform the disc consistency check:

       D/CCK n

    The response is

       1/<disc> 0% FULL

15. Re-IPL the System from disc (similar to step 6, but done only once), and again do steps 10 and 12 (but not 11). The DIRECT process is created; this performs another file consistency check and then starts the VOLUMS and SPOOLR processes.

16. The essential Subsystem and other user files can be transferred from a back-up tape (see VOLUMS command TRANSFER) after process SUBSYS and any prospective users have been accredited (see DIRECT command NEWUSER).

17. Log-on may now commence. Note that the SPOOLR process will at this stage be configured for only JOURNAL and LP queues. It will be necessary to specify a CONFIG file using the S/CONFIG command, and to start and re-start SPOOLR before normal service can be offered.

Appendix: 2960 IPL

IPL is done from the ECP:

autoload	DO ALD pts	(Kent 2960)	send twice if
	DO AL pts	(Dalkeith 2960)	IPL from tape
register load	RT		
run	GO		

To set the handkey register:

SW <hex value>

To halt the OCP (and display current PC)

HA (use GO to restart)

Hardware dump:

DO DUMP pts	(Kent)
DU pts	(Dalkeith)

EMAS 2900 dump:

Halt	HA	
Reset	RT	
Set handkey	SW ptsm	port trunk stream mechanism (of dump tape)
Run	GO	

The image store may be read and written to (if appropriate) from the ECP if you know the equivalent "working store" word numbers (engineers will tell you):

RWS <word>
WWS <word> <value>

The OCP control register #6011 is WS #27.

J. Maddock
(revised by P. Stephens
and J.K. Yarwood)

No: 7
Date: 11/12/79

DBOOT

DBOOT is the piece of code entered immediately after a disc IPL. It is concerned with:

- re-arranging and tidying main store
- on dual systems, IPL'ing the other OCP
- entering CHOPSUPE

When the IPL button is pressed, a transfer into store is initiated from the device whose address is set up on the console.

1. the transfer starts at real address 0.
2. at the end of the transfer:
   - the OCP loads its microprogram from real address 64 (words) onwards, unless there is a 'stop' word there.
   - the image store locations in block 0 lines 0-15 are loaded from real words 32-47 and LSTB and PSTB are loaded somehow from words 48-63.
   - execution commences at the instruction addressed by PC.
   - whether PC and subsequent addresses are treated as real or virtual depends on the RAM bit in the SSR. If virtual, the PAGE and SEGMENT tables must already be set up and must also be able to address the controller communications area in words 0-7.
3. the communications area for the controller involved in the IPL is in real words 0-7, the response area being in words 6 and 7. There are no streams.
4. the OCP can establish SAC/TRUNK/STREAM of the input device by:

   SAC from the OCP interrupt flags in image store
   TRUNK from SAC interrupt flags
   STREAM from bits 24-31 of real word 6

The main store layout when DBOOT is entered, which it is vital to understand, is:

X'00000' communications area

X'00028' start of first hole available for code

X'00080' initial machine state, PC etc.

X'000D0' start of second hole for code

X'00100' OCP microprogram

X'13E00' start of third hole for code

X'14000' CHOPSUPE - Part 1

X'1C000' space for OCP microprogram overlays(P3 only)

X'20000' CHOPSUPE - Part 2

X'37000' limit of store referenced by the IPL, beyond this point the store may have parity errors and can be cleared only by writing zero quads

Start by determining the OCP type. The various image store addresses differ for P2/3, P4/0 and P4/>0. CTB is loaded with the address of the appropriate set.

If OCP is not an S1, pick up real word 6 (CRESP0) and check that the Normal Termination bit is set (*** else IDLE 'F000', not a normal termination) and save STREAM in MARK.

On a P4, set image store location 4012 bits 26-31 to allow SAC interrupts when the System is running.

Pick up the System Interrupt Register and extract the interrupting port number P, form the address of the trunk flags, X'44P00000', and read and clear the SAC interrupt flags. Extract the trunk number, T (*** else IDLE 'F001', no interrupt flags). Set PAW to X'PT0000'. After a tape IPL, MARK and PAW are both set to -1 (see SYSTEM NOTE 8).

If it's a dual OCP, do the remote IPL by sending a special message to the other OCP having first, if it's a P2/3 dual, NEQ image store location X'600A' with X'80' >> HISPORT !!!! Execute a busy loop for a while to give the other OCP time to do its IPL.

Clear the top end of SMAC 0 with long long zero words (X'37000'- X'40000')

If a P3, the OCP microprogram overlays are moved from wherever they are to X'4000'-X'8000' and the OCP's descriptor is updated appropriately. If it's a dual P3, the other OCP's descriptor is updated also.

The final step is to move the two parts of CHOPSUPE down to X'8000' - X'10000' - X'27000' and ACTIVATE it. (CHOPSUPE has to be split to accommodate the OCP microprogram)

A. Gibbons

No: 8
Date: 12/12/79

IPL'ing from Tape

Tapes to be used for IPL are written by VOLUMS and consist of:

- an extended label block; this contains the special 'stop' word at byte X'100' so that, if this block is read, the OCP stops in an orderly fashion
- a TIPLBLOCK
- a tape mark
- several 4K blocks (not written by VOLUMS but by BULK MOVER) comprising the 'system' to be IPL'd

This set of 4K blocks contains the same information as would have been read from disc in a disc IPL. The function of the TIPLBLOCK therefore is to read in these blocks and position them in store as if they had been read by a disc IPL, and then to start execution of this code, once again as though after a disc IPL.

The function of the TIPLBLOCK program in more detail is:

- clear mainstore (this program occupies the first X'1000' bytes)
- if it's a P3, move the OCP microprogram overlays to X'3A000' - X'3E000'
- move this program to the 'top' of store (X'3E000'), re-enter it there and initialise variables
- initialise a trace area
- compute PORT, TRUNK and STREAM of the IPL device
- load GPC microprogram and connect the IPL stream
- locate the first operable deck on the IPL stream
- read blocks from the tape, the first block going temporarily to X'38000', until either a tape mark is read or the next block read would overwrite the first block in its temporary home
- rewind the tape
- move the first block down to the bottom of store
- ACTIVATE the program that has been read in

Program errors

If a fatal error is detected, the program IDLEs:

F001 CRESP0 zero on entry

F002 not a normal termination

F003 SAC interrupt flags, which give the TRUNK, zero

F004 first four decks on the IPL stream are inoperable

F005 GPC microprogram load failed

F006 CONNECT IPL stream failed

F007 Controller Detected Error when reading tape

F008 more than 10 attempts to read a block

F009 backspace failed when retrying read block

F010 rewind fails

F011 sense fails

F012 MARK < -1

F013 non attention response received when an attention expected

Background notes

TIPLBLOCK is produced from a MAPLE program, source TBOOT1S. When this has been compiled and converted to EMAS object format, with the name ERCC08.SIPLF, it may be converted to a

```
%CONSTINTEGERARRAY TIPLBLOCK(0:n)
```

in file ERCC08.TIPLBLOCK by calling the command MAKETIPLBLOCK, source MAKETBS. The OBEY file TBOOT does all this.

A. Gibbons

No: 9
Date: 12/12/79

Using MAPLE

ICL's MAPLE (or SFL) compiler has been made available on EMAS by Bob Eager of Kent University. Although the use of an assembler, such as MAPLE, is not normally recommended, there are times when it is necessary (e.g. IPL routines) and MAPLE provides a good framework for writing 2900 machine code (e.g. 'natural' expression of the various operand formats, pseudo OP codes for versions of jumps and macros - very reminiscent of System/360 Assembler in fact).

To use the MAPLE compiler, type the following:

```
INSERT(ERCC08.MAPLE)
```

You need only ever do this once; it will be remembered in subsequent logons.

To enter the compiler, type:

```
MAPLE(sourcefile, OMF=Z.omffile, PLT=YES, KEYS=NO)
```

Note that the placement field in the OMF parameter must appear but is ignored.

There are some subsidiary commands, used to set up options before running MAPLE:

MAPLENOLIST	- prevents a listing file being produced.
MAPLELIST	- causes a listing file, called T#LIST, to be produced. This is a convenient name for many commands; see SEND, LOOK, etc. This is the default setting.
MAPLELIST(name)	- nominates a file to receive the listing.
MAPLELSIZE(size)	- sets maximum size of listing file (bytes). Default is 1000000.
MAPLEEDIT(file)	- nominates a file to be used for compile-time editing commands.
MAPLENEW(file)	- nominates a file to receive the edited source file.

The default is for no editing, and no source output.

All defaults may be restored by the command:

```
MAPLEDEFAULTS
```

The OMF output from MAPLE may be converted to EMAS load format by the command:

```
COMF(omffile,emasfile)
```

The MAPLE components are held in a PD file MAPLE0108. To rebuild the EMAS version of the MAPLE assembler, simply:

```
OBEY(MAPLE0108_MAKEITUP)
```

To make a traced version, use:

```
OBEY(MAPLE0108_MAKETRACE)
```

Summary of pdfile members

COMF2S	- source of OMF to EMAS load format converter
HELP	- this file
KDIAG3S	- source of diagnostic support routines for traced version
KMAPL	- object code of MAPLE assembler (EMAS format)
MAKEITUP	- OBEY file to build production version
MAKETRACE	- OBEY file to build traced version
MBASE3S	- source code of interface program
OMFSQ2S	- source of support routines for OMF converter

A. Gibbons

EMAS 2900 SYSTEM NOTE

No: 10
Date: 27/05/80

IDLEs at IPL

If certain errors occur before an Operator console has been found CHOPSUPE cannot tell the Operator. Instead it halts with an IDLE instruction. These signify as follows:

a) IDLEs during Booting (from Disc or Tape)

F001	CRESP0 zero on entry
F002	Not normal termination
F003	SAC interrupt flags, which give trunk zero on entry
F004	First 4 decks on IPL stream inoperable
F005	GPC microprogram load failed
F006	Connect IPL stream fails
F007	Controller detected error when reading tape
F008	More than 10 attempts to read block
F009	Backspace failed when retrying read block
F010	Rewind fails
F011	Sense fails
F012	MARK < -1
F013	Non attention response received when attention expected

b) Errors detected during GROPEing for devices

B00B	OCP not 2960, 2970, 2980, 2972 or 2976
00DD	No GPC found in configuration
0DDD	No Controllers found at all
FF00	Too many GPC's (>8)
FF01	Too many slots (>256) or supplied table too small
FF02	Too many entries in RESPONSE array
FF03	Too many magtape streams (>32) (in FORM TABLES)
FF04	Too many Operator console streams (>7) (in FORM TABLES)
FF05	Supplied table too small (in CHECKLIM)

c) Errors detected after GROPE completed

AAAA	Normal CHOPSUPE idle (awaiting command from Operator console)
3333	IMP %STOP executed (software error)
E00E	Dump to tape completed successfully
12121	Dump to tape failed (deck not known or faulty)
CCCC	Attempt to return from procedure invoked by Activate

d) Unexpected interrupts in CHOPSUPE

00F0	System error interrupt occurred (probable OCP fault)
00F1	External interrupt occurred (none ever expected)
00F2	Multi-processor interrupt occurred (none ever expected)
00F4	Virtual Store interrupt occurred (none ever expected)
00F6	Program error interrupt occurred (software or hardware fault)
00F7	System Call interrupt occurred (no System Calls ever made!)
00F8	OUT interrupt occurred (no OUTs in code!)
00F9	Extracode interrupt occurred (none ever expected)
00FA	Event pending interrupt occurred (none ever expected)
00FB	Instruction Counter interrupt occurred (always masked)

P.D. Stephens

Utilities

DUMP Routine, Version 26C

17/4/80

Version 26C of the dump routine is now in service on the ERCC 2970. The object file is ENGINR.ENGPRGY_DUMP26CY; it is inserted in ENGINR.SS#DIR. The entry points are READDUMP and PRINTDUMP.

The changes in 26C with respect to the previous version (26) are as follows:

* Command: READDUMP(dumptape,file,out)

  file default is DUMPFILE
  out default is .LP

  Command: PRINTDUMP(file,out)

  out default is .LP

* The initial option-selection prompts have changed. If the response to "DUMP ANALYSIS:" is "Y" or "YES" then the following two extra options are given:

  PHOTO: "Y" or "YES" causes a dump of the photograph to be output, if available.

  AMT: "F" or "FULL" causes the AMT to be printed, including zero entries.
  "Y" or "YES" causes the AMT to be printed, excluding zero entries.
  Any other response suppresses the printing of the AMT.

  If the response to "DUMP ANALYSIS:" is "F" or "FULL" then this is taken to imply responses to "PHOTO:" and "AMT:" of "YES" and "FULL" respectively.

* The Oper output has changed, to reflect the new Oper software.

* Executing processes are printed out before the Kernel queue. They are still referred to ("EXECUTING") in the Kernel Q, Run Q1 and Run Q2 outputs.

* In the output of the AMT, the letters B, D and S are used to specify the position of an epage:

  B - both in store and on the drum. Store table index given.

  D - on the drum. Drum table index given.

  S - in store. Store table index given.

* READDUMP now takes account of segment boundaries when extending the dump file.

John M. Murison

Formatting and Labelling Routines in ENGINR

The Engineer process ENGINR has 2 utilities concerned with initialising discs. A new disc pack must first be formatted and then labelled before a file system can be established on it.

The procedure for formatting is as follows:

1) Log into ENGINR (password is ENGC).

2) Mount the new disc on a spare drive and wait for Operator Station to report:

```
        O/EDnn  LOADED  NO  LABEL
    [or O/EDnn  LOADED  xxxxxx  FRGN if from VME/B]
```

3) Type

```
        FORMAT (EDnn)
```

taking great care that the drive mnemonic is copied correctly.

4) The program prompts for lower and upper cylinder and track limits.

A lower cylinder of -1 indicates all cylinders.
A lower track of -1 indicates all tracks.

Formatting proceeds at about 25 cyls/min on a lightly loaded system.

Labelling must be done after formatting. The procedure is:

1) Log into ENGINR

2) Type DLABEL(EDnn)

3) Reply to prompt

```
    IPL OR NORMAL:
```

as appropriate. IPL labels are only needed for system (as opposed to user) discs.

4) Reply to prompt

```
    6 CHAR VOLID:
```

with the volume name.

5) After "Labelled OK" is output on the terminal check with Master Oper that the new label has been read successfully.

P.D. Stephens

EMAS 2900 UTILITY NOTE

No: 4
Date: 9/2/78

Transferring files from and to the 4-75

1. The utilities VTAPE and READVTAPE described below must be run in a privileged process on System 4 EMAS - usually 'UTILTY'. This restriction will be removed eventually and it will be possible to run them from your own process. The tapes written by VTAPE can be read on the 2970 using the VOLUMS TRANSFER command. The READVTAPE routine can be used to READ files from 2970 back-up tapes into the 4-75 file system.

2. VTAPE(tape[,start chapter])

   If 'start chapter' is omitted the file will be written starting at chapter 1. The 'start chapter' parameter can be used if it is necessary to add files to a tape which already contains some files. The command prompts for FILE: until the reply '.END' is given. The format of the response should be either

full filename	e.g. ERCC06.BASE11S
or full filename(new full filename)	e.g. ERCC06.BASE11S(ERCC27.BASES)

   The second form can be used to give the file a different name on tape, and hence on the 2970. Note that the filename can be a single member of a partitioned file, but in this case a 'new full filename' must be provided because an individual member cannot be created on the 2970; e.g.

   FILE:ERCC06.PD8_VTAPE04S(ERCC06.VTAPE)

3. READVTAPE(tape)

   The command prompts for chapters and can be terminated by replying '0'. Each chapter specified is read into the process being used and is given the same filename as it has on the tape, truncated to eight characters if necessary, and without the user name of course. After using this command the file should be OFFERed to its owner. This last requirement will be unnecessary when the utility can be run in the owner's process. The 2970 back-up tape must be mounted without a ring and should be returned promptly to the 2970 machine room. There is a line printer listing giving chapter numbers and file names, kept near the back-up tapes.

Roderick McLeod

EMAS 2900 UTILITY NOTE

No: 5
Date: 9/2/78

Input of Batch Archive Tapes to EMAS

1) In process 'UTILTY' on System 4 EMAS give the following command:

COMMAND: RUN(ERCS04.BATPIN2Y)

The program responds by giving the prompt

TAPE:

When the tape has been specified, the program outputs the name of the first file on the tape, then prompts

EMASFILENAME:

You should reply either:

filename

or REJECT (the program then goes on to the next file, if any)

or NO (the program exits to command level).

The output file is a sequential file containing EBCDIC OMF records. It is suitable as input to the OMF loader as it is.

---

2) If the file contains a microprogram, then to get a source file suitable for edit-insertion (as data in a constintegerarray) do the following:

INSERTFILE(ERCS04.OMFTOSY)

OMFTOSRC(infile, outfile)

The size of the array, in bytes and in decimal words, will be given. It is left to the user to put in the appropriate declaration.

---

3) If the file contains source, then to convert it into an EMAS source file do the following:

INSERTFILE(ERCS04.BAPPY)

BAPP(infile, outfile)

The output file is a normal EMAS source file.

A. Anderson

No: 5 (Re-issue)
Date: 21/11/78

Importation of Software from VME/B and VME/K

We can accept software from VME/B which has been written to a magnetic tape using procedure "BARCHIVE", and from VME/K via procedure "COPY OUT". There is no limit on the number of files per tape, but libraries must be written out as separate subfiles.

To obtain the facilities described below, the following should be specified:

```
OPTION (SEARCHDIR=ERCS04.OMFDIR)
```

---

1) To input files from a VME/B "BATCH ARCHIVE" tape, log into UTILTY and give the following command:

   Command: READBTAPE(tapename, filename, pdfilename)

   The parameters which the user specifies indicate which of three methods of removing files from the tape is to be used:

   a) To search the tape for a particular file.

      Command: READBTAPE(tapename, filename)

      If the filename specified is found on the tape then the prompt

      EMASFILENAME:

      will appear. Reply with the name you wish the file to have on EMAS 2900. The file will then be read in and control returned to command level.

   b) To inspect the tape file by file, transferring or skipping files as required.

      Command: READBTAPE(tapename)

      The program responds by printing the name of the first file on the tape, and prompting:

      EMASFILENAME:

      You should reply either:

```
    filename
or  R         (the program goes on to the next file, if any)
or  STOP      (the program returns to command level)
```

c) To input all the files from the tape into one or more partitioned files. This is to enable large numbers of files to be input simply.

Command: READBTAPE(tapename,,pdfilename)

The program will ask if the files are to be converted into EMAS 2900 source files as they are read in (answer Y or N). Then the program will ask for a maximum number of files to be read into the partitioned file. If this number is reached before the end of tape, then a new partitioned file can be set as destination. Thus a tape containing more files than could conveniently be kept in a single partitioned file can have its contents spread over any number of partitioned files.

On detection of a double tape mark, the message:

TAPE ENDS

is output and control returned to command level.

---

2) To input from "COPY OUT" VME/K tapes, log into process UTILTY and give the following command:

Command: READKTAPE(tapename)

and respond as to READBTAPE (method (b) above).

---

3) If the file contains a microprogram, then to get a source file suitable for edit-insertion (to be data for a constintegerarray) do the following:

Command: CMP(infile, outfile)

The size of the array, in bytes and in decimal words, will be given. It is left to the user to put in the appropriate declaration.

---

4) If the file contains source, then to convert it into an EMAS 2900 source file do the following:

Command: CS(infile, outfile)

If the file contains ISO use a third paramater, ISO, to avoid translation.

---

5) If the file contains an OMF program which has no unusual structure or run-time requirements, then it may be possible to convert it into an EMAS 2900 object file. Do the following:

Command: COMF(infile, outfile)

Note: References beginning "ICL9CEZ" or "ICL9CM" are changed to "S#" and "M#" respectively.

If you wish to see an analysis of the OMF structure then first do the following:

Command: OMFPARM(MAP)

---

6) If the file contains an OMF program which is unsuitable for conversion, it may still be possible to run it on EMAS 2900 by using the OMF loader. Note that conversion is preferable since the OMF loader is more limited and less efficient than the standard loader. See me for details.

---

7) If an OMF file contains unsuitable references, then they can be renamed by doing the following:

Command: AMENDOMF(input, output)
ACTION:

Reply STOP in order to exit to command level
or
RENAME/ref.name=new ref.name (with no spaces)

All occurrences of the reference will be traced and amended.

Please report any difficulties.

A. Anderson

Magnetic Tape Utility Interface Routines

The routines described in this note provide a common interface with a series of routines available on 4-75 EMAS. They are designed primarily for "non-standard" tape utility programs, e.g. programs for writing or reading tapes for other operating systems. Several such programs already exist on the 4-75 and it should be possible to move them to EMAS 2900 with the minimum of alteration. The routines make no assumptions about the structure of information recorded on the tape.

externalroutine OPENMT (string (7) VOL)

Claims the tape with volume label VOL. If an asterisk is appended to the volume label then the tape is to be be mounted with a write permit ring; otherwise it is to be be mounted without a ring. Currently the tape must have a standard EMAS 2900 volume label (identical to 4-75 EMAS or IBM O.S. volume label). In due course VOLUMS will include a facility to allow the operator to give a temporary name to a tape with some other format of label. When the tape is claimed it is left at BT.

externalroutine UNLOADMT

Releases the tape and unloads it.

externalroutine READMT (integer AD, integername LEN, FLAG)

Reads the next block from the tape. On entry:

AD = address of an area into which to read
LEN= maximum length of read area

On exit:

FLAG = 0 successful
= 1 tape mark read
> 1 failure (details later)

If FLAG=0 then LEN = length of block read

externalroutine WRITEMT (integer AD, LEN, integername FLAG)

Writes to tape a block of length LEN from address AD. A non-zero flag on return indicates failure (details later).

externalroutine WRITETMMT (integername FLAG)

Writes one tape mark. FLAG non-zero on return indicates failure.

externalroutine SKIPMT (integer N)

Skips N blocks - forwards if N>0, otherwise backwards. For the purpose of this routine a tape mark is treated as if it were a block.

externalroutine SKIPTMMT (integer N)

Skips N tape marks forwards or backwards according to the sign of N.

externalroutine REWINDMT

Rewinds tape to BT.

Notes

These routines have been deliberately written with a simple functional interface. There is an obvious limitation (by the lack of a logical channel number) to their use for single tape jobs. The lack of failure flags for the majority of them is intentional in order to simplify their use.

Their specification has remained virtually unchanged over 8 years and has provided a convenient interface for a large number of programs for analysing, reading and writing non-standard magnetic tapes. Only the following changes are needed for programs being moved from 4-75 EMAS:

* it is essential to set LEN before calling READMT
* WREOFMT has been renamed WRITETMMT since this is consistent with the names of the other routines

Roderick McLeod

No: 8
Date: 21/7/78

Tape Copying Utility: COPYTAPE

This command is used to make a copy of a complete magnetic tape. The routine makes no assumptions about the contents of the tape except that two consecutive tape marks are taken as an indication of the end of useful information. The command takes no parameters.

It prompts

INTAPE:	reply with VOL label of input tape
OUT TAPE:	reply with VOL label of output tape.
MAX BLK LENGTH:	reply with length of longest block on tape.
COPY LABEL?	reply: Y. if you want the output tape to be given the same name as the input tape N. if you want the output tape to retain its current label.

The program prints the number of blocks (including tape marks) copied. Failure messages should be self explanatory.

Rodgrick McLeod

Utilities for Creating Supervisor, Director and Subsystem Files

Components of the operating system are object files which have been post-processed to reduce loading them to a trivial operation. This post-processing is carried out by one of the following:

SUPFIX for Supervisors

DIRFIX for Directors

SPFIX for System Process Basefiles, e.g. for VOLUMS and SPOOLR

SSFIX for the Edinburgh Subsystem Basefile and any others based on it, e.g. currently JOBBER

In some cases further processing is required - see the notes on individual commands, below.

Parameters

All four commands take two obligatory parameters and a third optional one. The first parameter should be the name of the object file containing the compiled version of the component. This will normally have been produced by linking a number of separate components using LINK. The second parameter is the name to be given to the output file. The third parameter can be used to specify an output file or device for diagnostics and a map of the "fixed" file.

Action of routines

The action common to all the routines is as follows:

* The input file is copied into the output file.

* The GLAP in the output file is modified according to information in the load data. Any external references are satisfied from within the file by filling in the PLT descriptor in the GLAP with the value it would need to have when in use.

* Relocation of values in the GLAP is carried out, again with respect to values of the start address of the relevant areas when the file is being used.

* Any unsatisfied references are filled with the ISO characters 'NORT' followed by the first four characters of the unsatisfied entry name. The effect is that if an attempt is made to call this routine the resulting dump will show the characters in DR and thus facilitate locating the fault.

* A map of the entry points is printed.

Individual Routines

The notes below refer to each of the variants of the FIX routine.

SUPFIX

The input should be a Supervisor object file with an entry point called "ENTER" at which it is to be entered after being loaded by CHOPSUPE. After being fixed, the header has the following format:

0 total length	1 start of code	2 physical size	3 type=10
4 reserved for sum check	5 date and time	6 start of GLAP	7 offset of PLT descriptor for "ENTER"

Note that the LOAD DATA is removed from the file and that the GLAP is aligned on a page boundary.

DIRFIX

The input should be an object file containing the Director. At the beginning there must be the current version of the primitive local routine DIRLDR. This must be the first routine in the link list; it is an assembler routine which is entered by the Supervisor when the Director is started and which copies the GLAP into the GLA file and then enters Director at routine "DIRECTOR". After the file has been fixed, as described above, the load data is removed and a call is made to routine BUILDSCT to add the System Call Table to the end of the fixed file. This routine is described elsewhere. The header of the final Director file is:

0 total length	1 start of code	2 physical size	3 type=11 (fixed Director)
4 reserved for sum check	5 date and time	6 start of GLAP	7 start of SC Table

Note that as a temporary arrangement word 3 contains a copy of word 7 (start of SC Table) and word 4 contains a PC relative jump of 8 halfwords.

SPFIX

This routine is used for fixing basefiles for System processes such as SPOOLR and VOLUMS. The input should be a linked object file with routine SSLDR as its first component. The initial entry to the basefile should be:

systemroutine SSINIT(integer MARK, ADIRINF)

The use of the two parameters will be described in a Director Note. After fixing, the output file has the following format:

0 total length	1 start of code	2 physical size	3 type=12
4 reserved for sum check	5 date and time	6 start of load data	7 start of object file map

As a temporary measure word 4 contains a PC relative jump of 8 halfwords to cause a jump to byte 32 of the file.

SSFIX

This routine,which is used for linking the Edinburgh Subsystem, is the same as SPFIX except that references to items in the System Call Table are satisfied at fix time. Every time a user logs in, a check is made to see that the version of the System Call Table currently operative is the same as that at the time the fix was done. If not the references are re-filled. After calling SSFIX it is necessary to process the Subsystem file further by calling the routine MAKEBASEFILE, described elsewhere.

R.R. McLeod

EMAS 2900 UTILITY NOTE

No: 10
Date: 3/10/78

Utility for Writing Character Files to IBM format Tapes

A first version of WRITEIBMTAPE is available in UTILTY.

Command:WRITEIBMTAPE	
EMASFILE:	must be a character file
TAPE:	
DSN:	for file on tape
LABEL:	
RECFM:	FB only at present
BLOCKSIZE:	
RECORDSIZE:	

Failure or output messages should be self-explanatory.

Modifications coming:

* More than one file at a time; note that more than one file can be written on a tape using this version repeatedly and incrementing LABEL.

* Possibly VB as another format.

Note that long lines are truncated, and an appropriate warning is printed; for IMP source this is probably more useful than wrapping round.

R.R. McLeod

No: 11
Date: 9/10/78

Backup/Archive Tape Maintenance Programs

CHECKTAPE(parms)

where parms = null (in which case the tape is prompted for)

or = tape (the name of the tape to be checked)

or = tape,list (the tape name and an output file or device)

CHECKTAPE checks that the specified tape can be read, and optionally lists the contents of the chapter header of each file on the tape to the file or output device specified. It also checks the validity of each chapter header.

The checking can be abandoned by pressing ESC ("INT:") STOP.

COPYARCH(parms)

where parms = read tape, write tape (tape to be read, tape to be written)

or = null (in which case the read tape and write tape are prompted for)

COPYARCH copies the specified Backup or Archive tape to a new Backup or Archive tape, only changing the tapename in each chapter header. Full error recovery is tried when a page fails to copy. If a chapter header fails to copy, a dummy chapter is created on the write tape to replace it. This keeps the chapter numbers in correspondence. If a page in a file fails to copy it is written with what ever was actually read. If there are any failures in writing the copy is abandoned.

The copying can be abandoned by pressing ESC ("INT:") STOP.

These procedures can be accessed by inserting the two files

ERCC20.TAPECOP2Y and ERCC20.MAG3Y.

W. Laing

EMAS 2900 UTILITY NOTE

No: 12 (revised)
Date: 9/2/81

File System Maintenance Utilities

This document describes the utilities for the maintenance and repair of EMAS 2900 file systems available in release 7 of MANAGR's file system maintenance programs. The source is held in MANAGR.MANPRGS_MAINT07 and the object in MANAGR.MANPRGY_MAINTY. A description of each utility is held in the source file, which has been made VIEWable.

Note that for readability some of the command names have embedded spaces in the descriptions below. These must not be typed when the command is called if OPTION NOBRACKETS is selected.

The following is a complete list of the utilities, with brief descriptions and page number references to full descriptions, which form the bulk of this document.

Utility	Page no	Description
ARCHOFF	3	Marks a file to prevent archiving
ARCHON	3	Marks a file for resumption of archiving
BAD PAGES	3	Generates a table of bad pages on specified disc
CCKOUT	3	Lists data recording progress of file system consistency checks
CHECK DAS	3	For a given disc, generates a list of files containing sections whose disc addresses are less than a given disc address
COPY INDEX	4	Creates a duplicate file index, and copies all files contained in the original
CREATE LOG FILE	4	Creates a circular log file for a given user and sets LOGFILE field in user's index
DELUSER	4	Deletes a username from the system
DEREGISTER CLASS	4	Deletes a sequence of usernames
DIRLOG	5	Copies Director's monitoring file into a temporary file for further processing
ENV	5	Prints current Supervisor and Director versions and the discs currently on-line
FSYS START	5	Prints range of pages available for users' files on a given disc
GET	5	Reports value contained in specified field in user's file index
HOLES HIST	6	Generates a histogram of free areas and sections on a given disc
HOW FULL	6	Reports current fullness of a disc
HOW FRAG	6	Calculates degree of fragmentation on specified disc
INACTIVE USERS	6	Produces sorted list of processes not used since specified date
LIST NNT	6	A synonym for command USERNAMES
LOG OUT	7	Copies user's log file (see CREATE LOG FILE)

LOST FILES	7	List VOLUMS.LOSTFILES
MOVE INDEX	7	Moves an index to another or to the same disc, possibly changing its size
MOVE INDEXES	8	Moves user indexes except SPOOLR and VOLUMS from one disc to another
NEWUSER	8	Accredits a new username
REGISTER CLASS	9	Accredits a sequence of usernames
REFRESH FILE	9	Forces the resiting of a specified file
REFRESH FILES	9	Resites files which have sections with disc addresses less than specified disc address
REFRESH INDEX	9	Resites an index on the same disc with the same attributes
REFRESH INDEXES	10	Resites all indexes on a given disc, with the same attributes
RENAME INDEX	10	Renames an index to a new username
SET	10	Allows values in specified file index fields to be set to new values
SET H NOARCH	11	Sets NOARCH bit in user's #ARCH file
SET MSG EEP	11	Sets EEP to 11 in file descriptors of user's #ARCH and #MSG files
SET SSBYTE	11	Sets byte which is reserved for Subsystem use in the file descriptor of a file
SSDESTROY	11	Destroys a file or sequence of files
SSFFILES	12	Lists names and full attributes of files belonging to a given user
SSFILES	12	Lists filenames of a given user
SSFINFO	12	Gives a synopsis of a file's attributes
SSFSTATUS	13	Allows modification of given file's file descriptor
SSIPERMIT	13	Allows the setting of specific permissions to given file
SSNKB	13	Gives a synopsis of index size and usage
SSNOF	13	Gives synopsis of number of files for a given user
SSPERMISSIONS	13	Returns OWNP and EEP for given file and reports any whole index permissions
SSPERMIT	14	Allows general access permissions to given file
SSREMOVE PRM	14	Reverses the effect of SSIPERMIT
SSRENAME	14	Renames a file
SSTRANSFER	14	Transfers ownership of given file
TEST BAD PAGES	14	Tests pages flagged as bad on given disc; returns to the System pages successfully written to
USERNAMES	14	Lists the users and major attributes for one or all discs
WHATFILE	15	Reports the file to which a given disc page belongs
WHATFSYS	15	Reports the disc on which a given user's index resides

*** ARCHOFF ***

Operation: Prompts for <user>, <fsys>, <filename>.

Effect: Marks a file to prevent archiving.

*** ARCHON ***

Operation: Prompts for <user>, <fsys>, <filename>.

Effect: Marks a file for resumption of archiving.

*** BAD PAGES ***

Operation: Prompts for <Fsys (or -1)>

Effect: Generates a table giving details of the bad pages on <fsys>.

*** CCKOUT ***

Operation: Prompts for <fsys>, <output file or device>.

Effect: Copies the data recording progress of file system consistency checks, from the circular file maintained by the DIRECT process into a standard text file. The maximum filesize generated is 256K.

*** CHECK DAS ***

Operation: Prompts for <low page no> and <Fsys (or -1)>.

Effect: Calls 'TO DO ALL FILES' and for each file of each user on each fsys calls DGETDA and, if the disc address of a section is less than <low page no>, prints user.file .

*** COPY INDEX ***

Operation: Prompts for <user>, <fsys>, <new user>, <new fsys>, <new index size (Kbytes)>.

Effect: Creates a new index for <new user> on <new fsys>, unless one already exists. Index attributes (delivery, basefile, file limits etc.) are copied from the old to the new index. For each file in the original index, the corresponding file in the new index is destroyed (if it exists), a file of the same name and of the same size is created in the new index, and data are copied in from the original file. The file is given the same attributes as the original file.

Purpose: To run some important file system utilities (e.g. MOVE INDEX) following corruption on a file system, it is generally necessary to complete the file system consistency check first. In the case where corruption has occurred on the disc containing MANAGR (and hence the utilities also), this can be difficult. If COPY INDEX is used (say once per week, or after updates) to copy the contents of MANAGR's index to a second disc, then the utilities should be available to deal with the original disc, even if the original MANAGR index is inaccessible. In addition, the copied index forms an on-line backup of the original.

*** CREATE LOG FILE ***

Operation: Prompts for <user>, <fsys> and <logfile>.

Effect: Creates <logfile>, permits it to MANAGR, initialises <logfile> as a circular file and calls DSFI to set the LOGFILE field in <user>s index. The contents of the file are made available by LOGOUT (q.v.).

*** DELUSER ***

Operation: Prompts for <user> and <fsys>.

Effect: Deletes the file index for <user> on <fsys>, hence removing <user> from the System. This routine requires the user to confirm the username to be deleted before actually doing so. If <fsys> is given as -1 then all occurrences of the given username will be deleted automatically. If not then all other fsys's will be searched for other occurrences of the username. Any found will be reported and the user asked if they should also be deleted.

*** DEREGISTER CLASS ***

Operation: Prompts for a "base" username in which the last two characters are decimal digits (e.g. ERCC01), <fsys> and a number, N, of usernames to be deleted.

Effect: The "base" username and N-1 consecutive usernames (last two digits being incremented) are deleted from the System. (Equivalent to repeated calls of DELUSER, q.v.).

*** DIRLOG ***

Operation: No parameters

Effect: Copies the information from Director's monitoring file VOLUMS.#DIRLOG into a character file T#OUT which can then be further edited if desired.

*** DO ALL INDEXES ***

Skeleton routine, a copy of which can be used to surround code which is meant to be executed for all indexes on an fsys.

*** ENV ***

Operation: No parameters.

Effect: Prints the current Supervisor and Director versions, the process number of the calling user and the discs currently on-line. N.B. The first disc specified is the SLOAD disc.

*** FSYS START ***

Operation: Prompts for <fsys>.

Effect: Prints the range of pages available for files on <fsys>.

*** GET ***

Operation: Prompts for <what>, <user> and <fsys>

Effect: <what> is one of

ACR	ADDRTELE	AFILES
ARCHINDUSE	BASEFILE	BATCHSS
CODES	CONNECTT	CONTROLFILE
CUMINSTRS	CUMMSECS	CUMPTRNS
CURRPROCS	DELIVERY	DINSTRS
DIRMON	DIRVSN	FILES
FUNDS	GPFSYS	GPHOLDR
INDEXUSE	ISESSM	LASTLOGON
LOGFILE	MAXFILE	MAXKB
MAXPROCS	NKBIN	NKBOUT
PRIVILEGES	SESSINSTRS	SESSMSECS
SESSPTRNS	SIGMONLEVEL	SPECIALSS
STKKB	SURNAME	

Reports the value or values of the index attribute specified (some have multiple values associated).

For the significance of the index attribute, current Director documentation on the function DSFI should be consulted for all except PRIVILEGES. See SET for a full description of available PRIVILEGES.

*** HOLES HIST ***

Operation: Prompts for <fsys>.

Effect: Generates a histogram showing how many free areas there are of each size (1 to 32 Epages) on <fsys> and how many sections.

*** HOW FULL ***

Operation: Prompts for <fsys>.

Effect: Reports the current fullness (%) of the specified <fsys>. (The figure reported when the System is open to users is not strictly comparable with the corresponding figures given by the FCHECK process at System start-up, in that a considerable quantity of temporary file-space will be included in the figure for a running System).

*** HOW FRAG ***

Operation: Prompts for <fsys> (or -1).

Effect: Calculates the degree of fragmentation on specified <fsys>. The degree of fragmentation is defined to be the percentage of the total disc space available for user files which is occupied by sections which have sizes less than a full section (i.e. 1 to 31 Epages).

*** INACTIVE USERS ***

Operation: Prompts for <trigger date> in the format DD/MM/YY and <op file/dev>.

Effect: Produces a sorted table (oldest to youngest) of entries of the form date process last used, user, fsys, surname and delivery info of those users who have not accessed their process since the trigger date.

*** LIST NNT ***

This is a synonym for command USERNAMES (q.v.).

*** LOG OUT ***

Operation: Prompts for <user>, <fsys> and <file>.

Effect: Connects <user>s logfile, if it exists, and copies it to <file> (see 'CREATE LOG FILE').

*** LOST FILES ***

Operation: Prompts for <fsys>.

Effect: Lists the contents of VOLUMS.LOSTFILES on <fsys>.

*** MOVE INDEX ***

Operation: Prompts for <user>, <fsys>, <new fsys>, <new index size (Kbytes)>.

Effect: The program first checks <fsys> and <new fsys>. If these are different then the routine will operate in COPY mode, when copies are made of the files on <fsys> to <new fsys>. When <fsys> and <new fsys> are the same then the program can potentially operate in one of two modes: COPY or TRANSFER. In TRANSFER mode the files are not copied, ownership is merely transferred. This route is faster but less safe in that a crash occurring during file transfer will leave two partial indexes and possibly result in the loss of files. In practice the TRANSFER route is only obligatory if there is insufficient space on the disc to hold a copy of all the <user>'s files. The routine checks whether this is possible given the current state of the disc and if so offers <C/T?> to the caller otherwise the move can only proceed via the TRANSFER route.

After mode has been decided a new index of <new index size> named NEWZZZ is first created on <new fsys> and <user>'s index is renamed to OLDZZZ. The <user>'s index attributes are then copied to the new index, and all the files are transferred or copied from the old to the new index. If the mode was COPY then at this point a final chance is offered to the caller to abandon the move with no side effects. If the response is to continue or mode is TRANSFER then finally the OLDZZZ index is deleted and the NEWZZZ index is renamed to belong to <user>. Note that if the original index was corrupted, or if some files are marked as being in use (either because the owner's process had terminated in disorder or because the owner or other users are actually using the files), then the program reports the number of files it is unable to transfer and prompts "Continue?", before commencing the index move. If the reply is "N" or "NO" the move is abandoned, with no side-effects. If the reply is "Y" or "YES" those files which cannot be transferred are lost when the original index is deleted. In general it is preferable not to proceed with the operation if it is likely that one or more files are actually in use by a currently existing process. The file pages which are actually in use will not be re-used before the next IPL, but the process(es) using the file(s) will be unable to disconnect the files.

Purposes: One or more of the following:

1) To transfer as many files as possible from a corrupted index into a new index.
2) To move an index onto a different disc.
3) To change the size of an index.

Apart from possible software errors, indexes become corrupt mainly when a hardware error or machine stop occurs during updating of an index. Corruption is often first noticed during the file system consistency check at System start-up, when the message

```
<user> CORRUPT?  FSYS <fsys> or
<user> CELLS?  FSYS <fsys>
```

is given at the main OPER. In the former case the System remains closed to users until explicitly opened, preferably following a move (or re-creation) of the affected index(es). The latter message indicates that some list-cells are not attached to any list; correction is not normally urgent.

If the System is inadvertantly or otherwise opened and processes are started following a "CORRUPT?" message, the affected indexes should be destroyed and the users re-accreditted. Corruption of other users' files could otherwise occur.

*** MOVE INDEXES ***

Operation: Prompts for <from fsys> and <to fsys>.

Effect: Moves each user's index on <from fsys> to <to fsys> with the exception of SPOOLR and VOLUMS.

*** NEW USER ***

Operation: The program prompts for the following data:

Username
File system
Index size (Kbytes) (reply 4 or 8, for about 70 or 140 files respectively)
Initials and surname
Delivery information
Foreground password (4 chars)
Background password (4 chars)
Maximum total filespace (Kbytes)
Maximum single filesize (Kbytes)
Maximum process concurrencies allowed for the user, for interactive, batch and total numbers of processes.

Effect: A new user with the specified attributes is created.

*** REGISTER CLASS ***

Operation: Prompts for a "base" username (in which the last two characters are decimal digits), e.g. ERCC01, <fsys> and a number N of usernames to be accreditted. The program further prompts for:

File system
Index size (Kbytes) (reply 4 or 8, for about 70 or 140 files respectively)
Initials and Surname
Delivery information
Password (sets the 4 characters input as both foreground and background passwords)
Maximum total filespace (Kbytes)
Maximum single filesize (Kbytes)
Maximum process concurrencies allowed for the user, for interactive, batch and total numbers of processes.

Effect: The "base" username and N-1 consecutive usernames (last two digits being incremented) are accreditted to the System, all with the same process details as input initially. The program reports each username successfully accreditted. If any username cannot be accreditted (e.g. because the file system index area is full, or because the username already exists) the program terminates at that point.

*** REFRESH FILE ***

Operation: Prompts for <user>, <fsys> and <file>.

Effect: Forces a resiting of <file> owned by <user> on <fsys>.

*** REFRESH FILES ***

Operation: Prompts for <low page no> and <fsys>.

Effect: Will resite all files on <fsys> which have sections whose disc addresses are less than <low page no>. Used, for example, in reformatting a disc as a System disc.

*** REFRESH INDEX ***

Operation: Prompts for <user> and <fsys>.

Effect: Equivalent to MOVE INDEX with oldfsys = newfsys, old index size = new index size and MODE = TRANSFER. Resites an index on the same <fsys> with the same attributes. See MOVE INDEX.

N.B. Should not be done on the same <fsys> as the routine is running from.

*** REFRESH INDEXES ***

Operation: Prompts for <fsys>.

Effect: Does a REFRESH INDEX for each user on the <fsys> specified.

*** RENAME INDEX ***

Operation: Prompts for <user>, <fsys> and <newname>.

Effect: Calls 'DRENAME INDEX'.

*** SET ***

Operation: Prompts for <what>, <user>, <fsys> and <new value> (or <new values> as appropriate to the fields).

Effect: <what> is one of

ACR	ADDRTELE	AFILES
ARCHINDUSE	BASEFILE	BATCHSS
CODES	CONNECTT	CONTROLFILE
CUMINSTRS	CUMMSECS	CUMPTRNS
CURRPROCS	DELIVERY	DINSTRS
DIRMON	DIRVSN	FILES
FUNDS	GPFSYS	GPHOLDR
INDEXUSE	ISESSM	LASTLOGON
LOGFILE	MAXFILE	MAXKB
MAXPROCS	NKBIN	NKBOUT
PRIVILEGES	SESSINSTRS	SESSMSECS
SESSPTRNS	SIGMONLEVEL	SPECIALSS
STKKB	SURNAME	

Sets the new value or values of the index attribute specified. Some have multiple values associated and will prompt appropriately. For the significance of the index attribute, consult the current Director documentation for the function DSFI for all except PRIVILEGES, which are described below.

Note that in some cases it is not sensible and in other cases not permitted to SET certain attributes. See current DSFI documentation.

PRIVILEGES

It is possible to 'SET' PRIVILEGES or 'GET' PRIVILEGES (see 'GET', above). If the operation is GET then the individual PRIVILEGES enjoyed by <user> will either be listed in the form PRIVxx where 0 <= xx <= 31, or if none are enjoyed, given as *NONE*.

If the operation is SET then the current PRIVILEGES are first given as for GET, then the caller is prompted "PRIV:". This should be responded to by a reply of the form PRIVxx where 0 <= xx <= 31 (spaces are not significant). The caller is then prompted "G/R:"> and the reply determines whether the PRIVILEGE is to be given or removed. This sequence continues

until the reply .END is received to "PRIV:", when the new set of PRIVILEGES is reported. PRIVILEGES currently available are as follows:

PRIV 04	DPRINTSTRING, DDUMP
PRIV 07	DSFI 7
PRIV 08	DPERMISSION, DFINFO, DFSTATUS, DFILENAMES on other users' files
PRIV 09	DCHECKBPASS
PRIV 10	DSFI for privileged calls and other users
PRIV 12	Ability to reset BASEFILE, CONTROL FILE, TESTBASEFILE, BATCHBASEFILE
PRIV 15	Interactive use of magnetic tapes (DMAGCLAIM)
PRIV 17	DSFI 38
PRIV 18	DPON, DPON3, DOUT, DOUT11, DOUT18, DTOFF, DLOCK
PRIV 20	BADPAGE, DSYSAD, FBASE, GETAVFSYS
PRIV 22	ACREATE2, DMODARCH, DNEWARCHINDEX
PRIV 24	DCONNECT, DDISCONNECT on # files
PRIV 25	DPRG, DUNPRG, DTRANSFER, DOFFER
PRIV 26	DEMPTYI, DRENAMEINDEX, DNEWUSER, DDELUSER, VALIND, DDUMPI, DXDUMPI, GETUSNAMES
PRIV 31	Allows ADESTROY, DCHSIZE, DCREATE, DDESTROY, DNEWGEN, DRENAME to be used on someone else's file without full index permission.

*** SET H NOARCH ***

Operation: Prompts for <fsys> (or -1).

Effect: For each user on <fsys> sets the NOARCH bit in file descriptor of file #ARCH.

*** SET MSG EEP ***

Operation: Prompts for <fsys> (or -1).

Effect: For each user on <fsys> sets EEP to 11 in file descriptors of files #ARCH and #MSG.

*** SET SSBYTE ***

Operation: The program prompt is for <user>, <fsys>, <filename> and <values>.

Effect: For the specified file the "ssbyte" is set to (the rightmost 8 bits of) <value>. The "ssbyte" is a byte in the file descriptor reserved for exclusive use of subsystems.

*** SSDESTROY ***

Operation: The program prompts for <user>, <fsys> and then repeatedly for <filename> until a response ".END" or ".E" is given.

Effect: After each input <filename> an attempt is made to destroy <filename> belonging to <user>. A result code and brief text describe the success or failure of each "destroy".

*** SSFFILES ***

Operation: The program prompts for <user>, <fsys> and <output file or device>.

Effect: The result is exactly as for SSFILES (see below), except that extra details - largely as described under SSFINFO - are additionally printed for each file. "SSFFILES" stands for "SS Full FILES".

*** SSFILES ***

Operation: The program prompts for <user>, <fsys> and <output file or device>.

Effect: A list (in alphabetical order) of all the files belonging to <user> is placed in the <output file or device>.

*** SSFINFO ***

Operation: The program prompts for <user>, <fsys> and <filename>.

Effect: Prints out a concise synopsis of the file's attributes, namely:

"*" if "cherished"
"+" if marked for archive
<filename>
connect address in caller's virtual memory, or zero if not connected in same
physical size in epages
access permission to file owner (OWNP)
general access permission to all other users (EEP, "everyone else's permission")
access permission field (APF) from segment table, if file connected in caller's virtual memory, otherwise zero
current number of users of the file
list pool number to which descriptor cells belong in <user>'s file index
the user to whom the file is "on offer", if any
whether the file is marked:
PRIVacy VIOLated
TEMPorary
VTEMPorary
NOt to be ARCHived
as having more than one generation

The current Director documentation further explains this terminology.

*** SSFSTATUS ***

Operation: Prompts for <user>, <fsys>, <file>, <act>, and <value>.

Effect: Allows attributes of <file> belonging to <user> on <fsys> to be modified by calling the Director function DFSTATUS. N.B. This routine should NOT be used unless you are absolutely sure that you know what you're doing!!

*** SSIPERMIT ***

Operation: Prompts for <user>, <fsys>, <file), <to user> and <prm 1-15>.

Effect: Gives <to user> permission to access <file> owned by <user> on <fsys> as specified by the response to <prm 1-15>. If <file> is specified as .ALL then the appropriate whole index permission is granted. See also SSREMOVEPRM.

*** SSNKB ***

Operation: The program prompts for <user> and <fsys>.

Effect: The following data are printed from <user>'s file index:

Total file space (Kbytes)
Total temporary filespace (Kbytes)
Total "cherished" filespace (Kbytes) (currently reported equal to total filespace)
Maximum permanent filespace which the user is allowed to own (on-line)
Maximum single filesize allowed for <user>.

*** SSNOF ***

Operation: The program prompts for <user> and <fsys>.

Effect: The following data are printed from <user>'s file index:

Number of files
Number of temporary files
Number of cherished files
Number of file descriptors currently extant (this may exceed number of files, as some descriptors may be marked "dead", awaiting a garbage collect)
Maximum and currently free numbers of list cells in up to 4 list-cell pools.

*** SSPERMISSIONS ***

Operation: Prompts for <user>, <fsys>, <file>

Effect: Gives OWNP and EEP for <file> and whole index permissions. If <file> is specified as .ALL then whole index permissions only are given.

*** SSPERMIT ***

Operation: The program prompts for <user>, <fsys> and <filename>.

Effect: The specified file is given general access permission, i.e. execute, write and read.

*** SS REMOVE PRM ***

Operation: Prompts for <user>, <fsys>, <file> and <to user>.

Effect: Removes individual permissions to a file or index. Reverses the effects of SSIPERMIT.

*** SSRENAME ***

Operation: The program prompts for <user>, <fsys>, <filename> and <newname>.

Effect: The specified file is renamed to <newname>.

*** SSTRANSFER ***

Operation: The program prompts for <user>, <fsys>, <filename>, <newuser>, <newfsys> and <newname>.

Effect: <filename> belonging to <user> on <fsys> is transferred to ownership of <newuser> in <newfsys> and with name <newname>.

*** TEST BAD PAGES ***

Operation: Prompts for <fsys> (or -1)

Effect: Attempts to write to each page flagged as a bad page on <fsys>, by using the Bulk Mover to write successively a page of X'FFFFFFFF', a page of X'08CEF731' (the most difficult for the hardware) and finally the empty page pattern. If all of these are successful then the user is informed, the page removed from the bad page list and returned to the system. See BADPAGES.

*** USERNAMES ***

Operation: Prompts for <FSYS>, <SORT TYPE>, <USERNAMES> (if "all FSYS" option selected), and <FILE/DEV>.

Effect: A list of usernames accredited on <fsys> (if specified - otherwise all on-line file systems), together with leading file index attributes (file limits, process concurrency limits, file space etc.) is placed in <file or device>. If

the single FSYS option is selected then the sorting can be by index number or by username. If on the other hand all on-line file systems are selected the sorting may be alphabetic by username or surname or both. Optionally in this case a file called USERNAMES may be created or updated in the calling process. This file contains a directory of usernames sorted by surname.
Command LISTNNT is a synonym for command USERNAMES.

*** WHATFILE ***

Operation: The program prompts for <disc address> and <fsys>.

Effect: File system <fsys> is searched for the file(s), if any, to which epage number <disc address> belongs. If <fsys> is specified as -1 then all file systems are searched. The filenames, if any, are reported.

*** WHATFSYS ***

Operation: The program prompts for <username>.

Effect: Each on-line file system is searched for a file index belonging to <username>. Each <fsys> found to contain <username> is reported.

C. McCallum

No: 13
Date: 25/1/79

Selective Printing of a Hardware Dump Tape

The 2900 has a hardware facility to dump the first 256K bytes of store to magnetic tape as one block. (This should not be confused with the EMAS 2900 dump to tape facility.)

A program is available to read the block and dump selected parts of it to the line printer. Since the tape is unlabelled it is necessary to give it a pseudo-label as follows:

* Mount the tape on deck Mnn without a ring.

* When VOLUMS reports that the tape is mounted with no label, type:

```
V/Mnn=vol
```

  where vol is any 6-character volume label.

* Log in to ENGINR

```
Command: PRINTHWDUMP
TAPE: vol           as given above
CODE I/E?           ISO or EBCDIC interpretation
DUMP FROM BYTE:     reply first byte required
DUMP LENGTH:        reply number of bytes required
MORE:               reply Y to return to request a further area, or N to
                    terminate the run.
```

R.R. McLeod

Exporting Object or Source Files to VME/B and VME/K

EMAS 2900 object files for transfer to VME/B or VME/K must first be converted into ICL's object module format (OMF). They can then be written to tape in magnetic engineering format (MEF). This can be read at the target system by BPRIME or KPRIME.

To obtain the facilities described below, do the following:

```
Command: OPTION (SEARCHDIR=ERCS04.OMFDIR)
```

a) To convert an object file into OMF, ordinary users should do the following:

```
Command: COBJ (EMAS object file, OMF file)
```

System staff preparing ERCC products for release should use the similar procedure:

```
Command: OPUT (EMAS object file, OMF file)
```

Notes

* In each case the OMF file appears on EMAS 2900 as a sequential data file.
* A pd file member can be specified as input or output.
* Code entries beginning "ICL9CE" and system routines will be keyed.
* The module name given is the EMAS 2900 file name.

There are three small differences in the OMF module generated by calling OPUT from that generated by calling COBJ:

1) The "library" bit in the diagnostic record is set. This causes the module to be ignored by ICL's diagnostic trace-back.

2) "ENV " is set as the diagnostic "SUBNAME".

3) The module name is prefixed by "ICL9CE".

b) A procedure is available to control some aspects of the OMF generation. It should be called before execution of COBJ or OPUT, as follows:

```
Command: OMF PARM (control)
```

Some of the "control" settings are:

MAP produces a list comprising, the module name, entries and external references, together with the "IIN" assigned to each and their key strength.

MAXKEYS sets all code and data entries to have key strength. Note that this produces a module having the same "visibility" of names as on EMAS. It is not set by default because too many keys can be a nuisance, particularly on VME/K.

NOCASCADE means that when being loaded, the module will not attempt to satisfy references from the library list.

LIBPROC sets the "library" bit.

c) Writing files to tape.

Log into process UTILTY and do the following:

Command: WRITEBTAPE or

Command: WRITEKTAPE

The program will prompt for the tape name, and then accept an unlimited number of files, by prompting:

INPUTFILE:

To close the file, reply with "NO" or "STOP".

Source files are converted to EBCDIC variable blocked records by the program. If a pd file is specified then every member is written separately to tape, each identified by its member name only.

A. Anderson

EMAS 2900 UTILITY NOTE

No: 15
Date: 28/3/79

Magnetic Tape Utilities

The utilities described below have been transferred from 4-75 EMAS. They apply only to magnetic tapes which are 9-track, 1600 bpi, phase encoded, with odd parity and an inter-block gap of approximately 0.5-0.75 inch. Block sizes must be less than 12288 bytes.

To access these utilities:

Command: OPTION(SEARCHDIR=CONLIB.TAPEUTILS)

1. Tape analysis

a) Command: QUICKANAL(tapename, outdev) [alias SHORTANAL]

gives a block-by-block structural analysis of the tape.

tapename	will be prompted for if omitted
outdev	is .OUT by default

On reaching a double tape mark

Continue?
~~CONTINUE?~~

is prompted - reply Y or N.

b) Command: IBMTAPEANAL(tapename, outdev) [alias TAPEANAL]

assumes an IBM standard labelled tape, and gives a one-line analysis for each file on it.

~~tapename will be prompted for if omitted~~ stet
outdev is .OUT by default

2. Tape reading

a) Command: UNLABLDTAPE(tapename)

is used to read files from an unlabelled or non-IBM-standard labelled tape into an EMAS 2900 process.

~~tapename will be prompted for if omitted~~ stet

For each file to be read the program prompts as follows:

NEXTFILE NO:	absolute file number, or 0 to terminate the program
RECORD FORMAT:	F, FA, V, VA

RECORD LENGTH: or MAX BLOCKSIZE:	which of these appears is dependent on the choice of RECORD FORMAT (F/FA or V/VA respectively); reply with the number of bytes
CHARS/BINARY:	C or B; if C is given the program assumes the tape to be in EBCDIC and converts to ISO
EMAS FILENAME:	name of the file to be created
[EMAS RECFM:	F or V - only appears if RECORD FORMAT reply was 'F' or 'FA']
OFFER FILE TO:	recipient process, or '.' or ME if to be retained by you
START AT BLOCK:	usually 1, but may be otherwise if part of the file is not wanted or has been read previously

The resultant file is a data file, with structure

F(A) and record length given, or V(A) and record length 1024

containing either ISO characters or binary data. If the tape file would produce more than 1 Megabyte of EMAS 2900 file, then the program produces one file for each Megabyte or part thereof, named

filename, filenameA, filenameB, etc.

N.B. When V input format is specified, the whole magnetic tape block is assumed to be a single record - you must perform any unblocking needed.

b) Command: READIBMTAPE(tapename) [alias TAPETOEMAS]

is used to read files from an IBM standard labelled tape into an EMAS 2900 process.

For each file to be read the program prompts as follows:

DSN:	file name, or '.' if you do not know nor care; .END terminates the program
CHARS/BINARY:	C or B; C causes EBCDIC to ISO translation
EMAS FILENAME:	file to be created
OFFER TO?	recipient process, or '.' or ME if to be retained by you
SKIP TO BLOCK?	usually 1, but may be otherwise if part of the file is not wanted or has been read previously

The resultant file is a character or data file, according to the CHARS/BINARY reply. If it is a data file, its structure will be as acquired from the DCB information in the tape file header block. If the tape file contains more than 1 Megabyte of data the program produces one file for each Megabyte or part thereof, named

filename, filenameA, filenameB, etc.

3. Tape writing

a) Command: WRITEIBMTAPE(tapename) [alias STTOTAPE]

is used to write character files to an IBM standard labelled tape.

tapename will be prompted for if omitted

The program initially prompts as follows:

START AT LABEL:	specify first free IBM label number on the tape
DSNS=FILENAMES?	Y or N, depending on whether, for each file, the tape data set name is to be constructed from the EMAS 2900 name
CONSTANT RECLS?	Y or N, depending on whether LRECL is the same for every data set to be written
CONSTANT BLKSZ?	Y or N, depending on whether BLKSIZE is the same for every data set to be written

Then, for each file to be written to the tape, further prompts are as follows:

EMASFILE:	character file name; .END to terminate the program
[DSN:	only if DSN=FILENAMES? was N]
LRECL:	number of bytes - only ONCE if CONSTANT LRECLS?=Y
BLKSIZE:	number of bytes - only ONCE if CONSTANT BLKSZ?=Y

4. Tape dumping

Command:DUMPMT (tapename, outdev)

gives a character and hexadecimal printout of the contents of part of a magnetic tape.

outdev is .LP by default

The program prompts:

CODE I/E:	to select ISO or EBCDIC interpretation of the bytes on output.
SKIP:	to specify the number of blocks to be skipped before dumping.
BLOCKS:	to specify the number of blocks to be dumped.

Note that for the last two replies each tape mark counts as one block.

M.D. Brown
R.R. McLeod

EMAS 2900 UTILITY NOTE

No: 16
Date: 1/10/79

Procedures in Files CONLIB.SERV1Y and CONLIB.SERV2Y

Notes

* Routines intended primarily to be console commands (having string parameters) are marked "C" in the list below, and are described as commands in the text. Other routines and functions are given formal IMP specifications in the text.

* The column headed File indicates whether the procedure is in file SERV1Y or SERV2Y.

* Routines marked '*' in the following list should be regarded as unsupported or subject to change.

* Queries should be directed to the originator, J.K. Yarwood, ERCC Room 2013, ext 2647.

CONTENTS (in alphabetical order)

Procedure	File	Command	Synopsis	Item Number
AINFO	1	C	Gives access permissions	2
AWAIT	1	C	Suspends process without prompt	41
BEL	2	C	Issues BEL characters	42
BIN	1		Converts a string to binary	18
COMPARE	1	C	Text file comparison program	4
CONC	2	C	Concatenates text files	49
COPF	1	C	Copies a file	26
DELI	2	C	Sets DELIVER information	50
DETA	1	C	Creates and detaches a file	13
DUMP	1		Hex dump (virtual addresses)	25
DUMPFILE	1	C	Hex dump (from filename)	14
DUMPVM	2	C	Hex dump (from virtual address)	9
EBCDICDUMP	1	C	Hex dump (from filename), EBCDIC	16
EXFILE	2	C	Extracts text from a file	17
EXTRACT	2	C	Copies a pdfile member	53
FROMSTR	1		Non-failing FROMSTRING	10
HTOS	1		Integer to hex string	55
HXSTOBIN	1		Hex string to binary	20
ITOS	1		Integer to string	11
LI	2	C	Lists files to .LP	57
METR		C	METER info since previous call	6
MINIT		C	Init call for METR	5
NKB	1	C	Number of Kbytes on file	12
NOF	1	C	Number of files	40
NRCODE	1	C	De-assembles 2900 code	59
NWFILEAD	1		Gives address of new file	33
PDCHECK	2		Checks pdfile members	60
PDINSERT	2	C	Creates a PDfile member	52
PIM	2	C	Calls the IMP compiler	7

QINFO	1	C	Brief "fileinfo"	1
RDFILEAD	1		Gives address of read-mode file	30
RDINT	1		Interactive integer read routine	21
RDINTS	1		Interactive integer real function	67
RECODELINES	2	C	De-assembles into 2900 machine code	24
REPLACE	2	C	Replaces pdfile member	54
RSTRG	1		Reads a line into a string	19
SEARCHF		*	Searches for text in pdfile	69
SEPARATE	1	*	Separates command parameters	22
TIM	1	C	Time-of-day to console	3
TPFILEAD	1		Gives address of temporary file	32
TSEARCH	2	C	Searches for text in pdfile	70
TSEARCHALL	2	C	All occurrences of text in pdfile	71
UDERRS	1		Prints a Director error message	23
UPDATE	2	C	Prepares job to update pdfile	61
VAL	1		Validates specified VM for access	72
WRFILEAD	1		Gives address of write-mode file	31
YCOMP	1	C	Binary comparison of two files	38
YSEARCH	1	C	Searches a file for integer or string pattern.	37

---

1) QINFO(file1, file2, ...)

A quick FILEINFO on one or more files - gives a one-liner describing each file.

2) AINFO(file1, file2, ...)

Gives own, general and individual access permissions for each file.

3) **TIM**

Gives time of day.

4) COMPARE(file1, file2)

Compares the two text files file1 and file2 on a line-by-line basis, stopping and printing out differing lines (if any). When a difference has been found, a prompt ':' is given, and the following commands may then be typed.

A	Advance one line in each file and restart the comparison.
Mfn	Move, in file f, n lines (f=1 or 2).
Q	Quit
E	End (same effect as Quit).
Pf	Print the current line of file f (f=1 or 2).
PB	Print the current lines of both files.
GO	Restarts the comparison of successive lines.
Fftext	Find in file f the text "text" (f=1 or 2).
FBtext	Find in both files the text "text".

The commands are required to be typed in a strict format and are carefully checked before being effected. An error leads to rejection, with the message NO.

The program terminates with the message EOF1, EOF2 (end of file1 or file2 reached) or COMPARISON COMPLETE (end of both files reached simultaneously).

5) **MINIT**
6) **METR**

Gives CPU time, page turns and number of SVCS since the previous call of METR. MINIT should be called once to start with for initialisation.

7) **PIM(file, parm, parm, ...)**

Calls the IMP compiler to compile the source file which is the first (or only) parameter. If the filename is FILE, the object and listing files are taken as FILEY, FILEL. (If the filename ends with 'S', the 'S' is dropped before the 'Y', 'L' are appended. The filename must not have 8 characters unless the last is 'S'. If the filename contains a 6-character username or an 8-character pdfile name, the object and listing filenames are generated from the 8-character file name or member name).

In addition, the following keywords may be appended:

.NY	object file to be .NULL
.N	listing file to be .NULL (also adds PARM (NOLIST))
.OUT	error lines to interactive terminal
.LP	listing file copied to .LP
.LPD	(".LP and Destroy") listing file sent to .LP
X	object filename to have X rather than Y (or Z) as last character
NEWGEN	causes object file to have X as a last character and provided compilation is successful, the object file is NEWGENned onto the existing 'Y' (or 'Z') object file.

NOLIST
OPT
NOCHECK
NOTRACE
NOARRAY
NODIAG
MAP
STACK
PARMX
PARMY
DEBUG
MAXDICT

} cause corresponding PARM to be set.

When the compilation is completed, the PARM for the process is reset to the parms which previously obtained.

9) **DUMPVM**

Produces a hexadecimal dump of an area of virtual memory. Prompts for start address or segment number; if a segment number is given, prompts additionally for a relative start address within the segment. Then prompts for a relative finish address and for output file or device.

11) externalstringfn FROMSTR(string (255) S, integer I, J)

Operates analogously to the IMP intrinsic string function FROMSTRING, except that conditions leading to the run-time fault STRING INSIDE OUT instead simply return a null result.

11) externalstringfn ITOS(integer I)

Returns a decimal digit string representation of the integer I (string commences "-" if I negative).

12) **NKB**

Prints the following file index data for the process:

maximum file size allowed
total file space limit
total Kbytes of file space
number of files
total Kbytes of temporary file space
number of temporary files.

13) **DETA**

No parameters. Prompts ':' and accepts input lines for a detach file until an input line comprises:

%C when it detaches the file and returns to command level.

integer when it detaches the file for execution with that time limit (minutes) and returns to command level.

Q when it just returns to command level.

14) **DUMPFILE(file, relstart, relfinish, filedev)**

Places a hexadecimal dump from address "relstart" to address "relfinish" in file "file" into file "filedev". "Filedev" may be a filename or .OUT or .LP. The dump has 32 bytes per line (16 for .OUT) with graphic representations for alphabetic and numeric characters on the right. Unless "filedev" is .OUT, the output file contains the name of the input file as a heading. "relstart" and "relfinish" may be given as decimal or hexadecimal (preceded be X) addresses relative to the start of the file. Only the first N of the four parameters need be given (0<= N<=4), as the program prompts for successive parameters which are not given (or which are given incorrectly).

15) DUMPCODE(file, relstart, relfinish, filedev)

Operates analogously to DUMPFILE (q.v.), placing de-compiled assembler from "file" into "filedev". Identical to NRCODE (q.v.).

16) EBCDICDUMP(file, relstart, relfinish, filedev)

Operates analogously to DUMPFILE above, except that graphic representations are for EBCDIC byte values rather than ISO.

17) EXFILE(file, text1, text2, filedev)

Extracts lines from "file" starting with line containing first occurrence of "text1" up to and including line containing first subsequent occurrence of "text2". Extracted file is sent to .LP. Fast , with minimum page turns. If "text1" is null, extract commences at start of file. If "text2" is null, extract ends at end of file. (Both "text1" and "text2" may not be null). Multiple consecutive underline characters in "text1" or "text2" are replaced by single space. If fewer than three parameters are supplied in the command file, the program prompts for its further requirements.

18) externalintegerfn BIN(string(255) S)

Result is the value represented by the string of up to eight decimal or hexadecimal (preceded by X) digits. Error result is X'80308030'.

19) externalroutine RSTRG(stringname S)

Reads characters from the currently selected input stream and sets S to be the string of characters obtained (excluding the newline). (Will not deliver a null string, and keeps reading until a non-null line is obtained).

20) externalintegerfn HXSTOBIN(string(255) S)

Result is the value represented by the string of up to eight hexadecimal digits (preceded by X). Error result is X'80308030'.

21) externalroutine RDINT(integername i)

Reads the next decimal or hexadecimal (preceded by X) number from the user's terminal. This routine is eqivalent to the standard IMP READ routine except that: (a) it will not read reals; (b) it accepts hexadecimal numbers; (c) it does not fail "symbol in data" for invalid input, but gives an error message for faulty input, abandoning remaining input on the same line and indicating the last valid number read (if any) which was supplied to the program, and then prompts for further input.

22) externalstringfn SEPARATE(stringname S)

Separates the string S into substrings comprising things between commas. At successive calls of the function the result and the string S are set to the 'next' substring. Result is null when there are no substrings left. (A null substring (i.e. ', , ') in the original also terminates the set of substrings).

23) externalstringfn UDERRS(integer I)

If I is an error number returned by a Director procedure, the function yields a text interpretation of the number.

24) RECODELINES(file, start line, end line, filedev)

De-assembles (2900 machine code) to file or device "filedev" from an IMP object program in "file" compiled with line-number updating between source line-numbers "start line" and "end line" (or line numbers reasonably close to same if code for stated line numbers is not found). Prompts if no parameters supplied, or parameters supplied incorrectly.

25) externalroutine DUMP(start, finish, printstartaddress, columns)

Prints a hexadecimal dump from virtual address "start" to virtual address "finish" on the currently selected output stream. The dump has either 16 (if "columns" <=72) or 32 bytes per line with graphic representations for alphbetic and numeric characters on the right. Multiple successive zero lines are printed once only followed by "ZEROES".

26) COPF(file1, file2)

Operates analogously with the Edinburgh Subsystem command COPY except that the file header words are not inspected and the physical size of the output file (file2) is the same as the physical size of the input file (file1). (Intended for non-standard files.)

30) externalintegerfn RDFILEAD(string(15) file)

Connects "file" in the "most suitable/possible" mode for reading. If the connect request cannot be satisfied, output stream zero is selected, the Subsystem routine CONNECT error flag is printed with an English interpretation, and a result of zero is returned. Otherwise the result is the virtual address of the start of the file (segment-aligned).

31) externalintegerfn WRFILEAD(string(15) file)

Connects "file" in write-mode, creating the file (1 page, but leaving a 4-segment VM gap) if it does not exist. If the connect request cannot be satisfied, output stream zero is selected, the Subsystem routine CONNECT error flag is printed with an English interpretation , and a result of zero is returned. Otherwise the result is the virtual address of the start of the file (segment-aligned).

32) externalintegerfn TPFILEAD(string(15) file, integer PAGES)

Operates analogously with WRFILEAD (q.v.) except that the file is created with the attribute TEMPORARY if it does not already exist (that is, the file will not survive a logout or a System crash).

33) externalintegerfn NWFILEAD(string(15) FILE, integer PAGES)

Operates analogously with WRFILEAD (q.v.) except that if file already exists it is connected in "newcopy" mode (previous file contents are not to be read).

35) PRHEX(integer I)

Prints "integer" in hexadecimal representation on the currently selected output stream.

36) PROCT(integer I)

Prints the octal representation of the RH half of "integer" on the currently selected output stream.

37) YSEARCH

Prompts for the name of a file in which a "binary" search for a byte pattern is to be made. Prompts RELSTART and RELFINISH (which may be given in decimal or hexadecimal (preceded by X) representation) for search limits relative to the start of the file. Then the program prompts STR/SHORT/INT: to determine from reply (STR, SHORT or INT) whether (respectively) a multi-character pattern, a halfword pattern (halfword aligned) or a fullword pattern (halfword aligned) is to be searched for. Then it prompts respectively for a non-null string of characters, a 16-bit integer or a 32-bit integer as appropriate (the integer may be typed in decimal or hexadecimal (preceded by X) representation). The search then commences. NOT FOUND is printed if the pattern is not found; otherwise an area around the first occurrence of the pattern is printed at the teletype.

38) YCOMP

Used to compare two files ("binary" compare). Prompts for the two filenames and a relative start position (which may be given in decimal or hexadecimal (preceded by X) representation). Compares the two files up to the end of the shortest file or up to the first difference in content. Prints the relative address at which the comparison terminates, with the contents of the words which are not equal if termination is as a result of difference between the files.

40) NOF

Prints the number of files in the process owner's index.

41) **AWAIT**

Suspends process without a prompt string on the terminal, thus allowing operator messages to be printed while the process is asleep.

42) **BEL**

Sends about 7 BEL characters to the interactive terminal.

49) **CONC(file1, file2, .../outfile[, .NP])**

Concatenates the text files file1, file2, ..., placing the resultant file in outfile (which may be the same name as one of the input files). If the keyword .NP is appended, a new page in the output file is placed at the end of each input file.

50) **DELI**

Prompts for up to 19 characters of delivery information (which may contain spaces), centralises it in a 19-character string and calls DELIVER.

52) **PDINSERT(pdfile1_member1, file2_member2, ...)**

Calls COPY(member1, pdfile1_member1),
COPY(member2, pdfile2_member2).

(The members must not already exist in the pdfiles.)

53) **EXTRACT(pdfile1_member1, pdfile2_member2, ...)**

Calls COPY(pdfile1_member1, member1),
COPY(pdfile2_member2, member2), ...

(Files called member1, member2, ... must not already exist.)

54) **REPLACE(pdfile1_member1, pdfile2_member2, ...)**

Calls COPY(member1, pdfile1_member1),
COPY(member2, pdfile2_member2), ...

(The members must already exist in the pdfiles.)

55) **externalstringfn HTOS(integer I, PLACES)**

The result is a string length of PLACES giving the (least significant digits of the) hexadecimal representation of the integer I.

57) **LI (file1, file2, ...)**

Calls LIST(file1, .LP),
LIST(file2, .LP), ...

59) **NRCODE(file, relstart, relfinish, filedev)**

Operates analogously to DUMPFILE (q.v.), placing de-compiled 2900 assembler from "file" into "filedev".

60) **PDCHECK(pdfile1, pdfile2, ...)**

For each pdfile a comparison is performed between members and ordinary files with the same name, and a summary is printed. At the same time a file SS#DESRP is created containing data for the command UPDATE (q.v.).

61) **UPDATE**

Data in file SS#DESRP is used to create a file SS#DETAC to be OBEYed or DETACHed to update the pdfiles in the most recent call of command PDCHECK (q.v.). The program gives groups of filenames: first those which would be destroyed, then those which would replace members of the pdfiles. For each group a corresponding number of Y's and N's must be typed (if the number does not match the number in the group the input is ignored). Where Y's are given, a DESTROY/REPLACE command is put in the SS#DETAC file. When all Y's/N's have been supplied the SS#DETAC file is listed, and the prompt "DETACH/OBEY:" is given. The replies are:

Q	No further action
NOW	SS#DETAC is detached for NOW execution
NOW, integer	As NOW, integer specifying "seconds"
integer	SS#DETAC is detached, integer specifying "seconds"
OBEY	The prompt .LP/.OUT is issued; .LP or .OUT is typed, to give OBEY(SS#DETAC, .LP or .OUT).

67) externalintegerfn RDINTS(string(63) S)

The result is the next integer which would have been delivered by routine RDINT (q.v.), except that if S contains valid integers separated by spaces or newlines, the next integer from S will be delivered.

69) externalintegerfn SEARCHF(params)

Not supported.

70) TSEARCH (text, file)

Searches for "text" in "file". Multiple underline characters in text are replaced by a single space. If a null parameter is given, the program prompts for text (which may include spaces) and for files. Lines surrounding the first occurrence of "text" in file are printed. If "file" is a pdfile, all character-type members are searched, except that if (for example) member FILE1 and FILE1L both exist, the FILE1 member is not searched.

71) TSEARCHALL(text, file)

Operates analogously with TSEARCH (q.v.), except that all occurrences of "text" in "file" are printed.

72) externalintegerfn VAL(integer ADR, LENGTH, RW, PSR)

Validates the virtual memory described by ADR, LENGTH for access according to RW (0=read, 1=write). If PSR is zero, the validation is at the ACR level of the caller; otherwise PSR should be the program status register value for the validation.
Result = 1 access allowed
= 2 access not allowed

J.K. Yarwood

Tape Utilities

The utilities described in this Note are for processing tapes written to recognisable standards. The tapes that the user can process with these utilities are user tapes, not tapes used by the System for its own private purposes (such as archiving). Users should note that although user (owned) tapes can be processed on EMAS 2900, they should consider the relative merits of the archive system compared with the use of private tapes for long term storage of data.

It is intended to provide the following utilities and detailed documentation for them. Only those marked with an asterisk are described in the current version of this Note.

TAPEANAL*	Summarise contents of a tape.
COPYTAPE	Copy one tape to another or copy selected files from several tapes to one output tape.
COPYTOTAPE* COPYFROMTAPE*	Copy files between user tapes and disc and vice-versa, including the handling of character files.
TAPECONVERT	Convert a tape written to an alien but reconisable standard to the System (IBM) standard.
DUMPTAPE	Dump out selected parts of a tape.

The latter two utilities will only be of help to those with a good knowledge of tape and file structures.

The utilities have been designed as a set, giving a reasonably consistent appearance, and can be used from a foreground or background (batch) process with reasonable ease. However, it should be noted that access to user magnetic tapes from a foreground process is not normally permitted; consult your local Advisory Service for details.

The parameters are specified in a parameter list to the call on the utility required, or in response to prompts, but not a mixture of the two. Positional and keyword parameter specification can be mixed, the use of a keyword parameter implying movement to the position of that keyword parameter for the purpose of identifying subsequent positional parameters.

Default values can be invoked when using a parameter list by omitting the relevant parameter, and at any point all remaining parameters can be defaulted by terminating the parameter list. If the user elects to be prompted for parameters (usually in foreground) he may specify that a default for a given parameter is to be taken by replying with the stop (".") character to the prompt for that parameter. If he wishes that all subsequent parameters (including the current one) should be defaulted, he should respond with the slash ("/") character; no more prompts will then be issued. Note, however, that some parameters do not have default values.

COPYFROMTAPE

This utility can be used for copying files from tape to the EMAS filestore. The tape must have been written to the accepted System standard, i.e. the 'IBM' standard. The utility takes the following parameters, which can be specified positionally (in the order below), by keyword or in response to prompts. Normally only the first four parameters will be needed.

Note that if the output file already exists, it will be overwritten by this utility.

Command:COPYFROMTAPE(name, label, type, rename, skip, process, maxrecl, size)

Parameter	Default	Description
NAME	none	The name of the file on the tape.
TAPE	none	The name of the tape volume.
LABEL	1	The position of the file on the tape.
TYPE	DATA	This parameter determines the format of the output file: it has four possible values. DATA file will be copied as data file. CHAR file will be copied as a character file. SEE NOTE (1) CHARX as CHAR except that newline characters will not be added at the end of each record processed. SEE NOTE (2) CHARA as CHARX except that if the input file is labelled as having ASA format control characters in its records, then those characters will be interpreted as the file is copied. (Not yet implemented.)
RENAME	tape file name	The name to be given to the disc file if its name is to be different from that of the file on tape. This must not be a member of a PD file.
SKIP	0	The number of records in the tape file to be skipped before processing commences.
PROCESS	*	The number of records to be processed. If all records from the starting point (i.e. after any SKIPped records) are required then specify '*'.
MAXRECL	10000	This specifies the maximum length of record in the input file. If this is greater than 10000 then this parameter must be used to specify the largest record in the input file.

Parameter	Default	Description
SIZE	255	This specifies (in Kbytes) the upper limit on the size of the output file when being created by COPYFROMTAPE. This is a temporary upper limit (see the Edinburgh Subsystem command DEFINE).

Examples

To copy the first file called SAMPLE1 from a tape AB4567 to a data file of the same name:

```
Command:COPYFROMTAPE(SAMPLE1,AB4567)
```

To copy the seventeenth file called STRANGE from tape KT5432 to a character file called FAMILIAR:

```
Command:COPYFROMTAPE(STRANGE,KT5432,17,CHAR,FAMILIAR)
```

The last example could be specified using a mixture of keyword and positional parameters, thus:

```
Command:COPYFROMTAPE(TAPE=KT5432,17,NAME=STRANGE,TYPE=CHAR,FAMILIAR)
```

Notes

1) When copying to a character file, all trailing spaces in each record are removed. Also, if TYPE=CHAR is specified, a newline character is added to the end of each record processed.

2) If TYPE=CHARX is specified, any records consisting entirely of space characters would be lost, as all the spaces would be removed and no newline character would be output.

COPYTOTAPE

This utility can be used to copy character or data files from the EMAS 2900 file system to a magnetic tape, which must already have a volume label. All the files will be written in accordance with the accepted System standard, i.e. the "IBM" standard. The utility takes the following parameters, which can be specified positionally (in the order below), by keyword or in response to prompts. Normally only the first four parameters will be needed; only the first two must be specified.

Command:COPYTOTAPE(name, tape, label, format, rename, skip, process, blocksize, option)

parameter	default	description
NAME	none	The name of the file in the EMAS file system. If a PD file member is specified, the RENAME parameter must be used to specify a valid IBM filename (see note 5).
TAPE	none	The name of the tape volume.
LABEL	1	The position of the file on the tape.
FORMAT	VB200	The required tape file format.
RENAME	input file name	The name to be given to the tape file if its name is required to be different from that of the input file.
SKIP	0	The number of records or "lines" to be skipped before processing commences.
PROCESS	*	The number of records or "lines" to be processed. If all records from the starting point are required then specify '*'.
BLOCKSIZE	4096	The length of blocks written to the tape. If this parameter is defaulted the System will use a blocksize as near to 4096 as is consistent with the specification of FORMAT (see above).
OPTION	STOP	For character files: specifies what action is to be taken in the event of a "line" in the input file being longer than the maximum record length of the output file. It has three possible values: STOP Processing is abandoned SPLIT The "line" is split across two or more output records (not yet implemented) TRUNC The "line" is truncated

Examples

1) To copy a character file called COMPRESS to the fifth file on tape AB1234:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5)
```

2) If the above file were required to be copied in "card image" format with a blocksize of 2000 the call would be:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5,FB80,,,,2000)
```

3) A keyword could be used to specify the blocksize in example (2) above:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5,FB80,BLOCKSIZE=2000)
```

Notes

1) When a file is copied to label "n" on a magnetic tape, any file recorded at that label position and any files recorded at further label positions are DESTROYED.

2) Program source files being copied to tape for transfer to other installations should probably be in "card image" format, i.e. with FORMAT set to FB80.

3) When a character file is copied to a file on tape with fixed length records, all "lines" are padded with spaces.

4) Multiple newlines from a character file copied to a tape file with variable length records are recorded as single records consisting of a space.

5) To conform to the IBM file naming convention, names of files on private tapes must:

   (a) be no longer than 17 characters in total.

   (b) comprise one or more fields separated by periods, each field consisting of 1 to 8 alphanumeric (A->Z, 0->9) characters starting with a letter.

   e.g. FILEA
   SAMPLE1.GLASGOW
   G35A.STAT.LEVEL2

This utility is used to scan a tape and print out a summary of its contents, the thoroughness of the scanning being controlled by the LEVEL parameter. It should be noted that reading every block on a tape, especially when it holds a lot of data, is a relatively expensive process. Using the LEVEL parameter set to 1 avoids this problem. The other values of LEVEL are normally used for diagnostic purposes.

Command:TAPEANAL(tape, output, level, dumpout)

parameter	default	description
TAPE	none	The name of the tape to be analysed.
OUTPUT	.LP	Destination for the summary output from the analysis. This can be an output device or a file.
LEVEL	1	The type of analysis obtained. It takes three possible values, the first two applying only to tapes conforming to the accepted System standard: 1 A scan and extraction of information from the labels on the tape. This gives an efficient summary of the characteristics of all the files on the tape. 2 As 1 but including a scan of the files as well as the labels, thus checking for I/O errors and reporting the longest block in each file. This is less efficient than 1. 3 This option assumes nothing about the structure of the tape but simply gives a block by block summary of each physical file on the tape.
DUMPOUT	null	If a dump of the first 96 bytes of the first two blocks of each file is required then this parameter must be used to to specify the file for the dump output.

Examples

1) To obtain a simple summary of the contents of tape AB1234:

Command:TAPEANAL(AB1234)

2) To obtain a complete summary of the same tape, including a dump of the first 96 bytes of the first two blocks of each (labelled) file on the tape, and then to print the summary and dump together:

Command:TAPEANAL(AB1234, SUMRYFILE, 2, DUMPFILE)

Command:LIST(SUMRYFILE+DUMPFILE, .LP)

Notes

1) Although the headings on the output for a level 1 analysis are reasonably self explanatory, the following notes may help:

   Block Length: the maximum permitted size for blocks in the file.

   Label No.: the file sequence number that appears in the file label (this need not necessarily start at one for the first file on the tape).

2) For a level 2 analysis three extra fields are printed:

   Max Block: the length of the largest block actually found in the file.

   Byte Count: the total count of all bytes in the blocks of the file, excluding the file labels but including all 'red tape' for variable format files.

   Length: an estimate of the actual length of tape occupied by the file (including its labels) rounded up to the next foot. This assumes an inter-block gap of 0.6 inch.

3) The output from a level 3 analysis relates to the physical files on the tape. The output for each file consists of the physical file number followed by a number of fields, as follows:

```
BLOCKNO( COUNT * SIZE)
```

   where SIZE is the length (in bytes) of COUNT blocks starting at block BLOCKNO, and taking the first block on the tape as block one. E.g.

```
13(   9 * 1600)
```

   would mean that there are 9 blocks of 1600 bytes starting from and including block 13 on the tape.

4) Users who wish to restrict the output from this utility to 72 characters per line should prefix the value for the LEVEL parameter by the letter 'S'. This suppresses the output of some of the less important information. For example:

```
Command:TAPEANAL(AB1234,.OUT,S1)
```

B.R.P. Murdoch

EMAS 2900 UTILITY NOTE

No: 19
Date: 26/05/80

PROBE INDEX

Source: MANAGR.MANPRGS_PROBEINnnS

Object: MANAGR.MANPRGY_PROBEINDEXY

Originator: C. McCallum

Parameters: None

Operation: Prompts for <user>, <fsys> and <output to>.

Effect: Reads the file index for <user> from <fsys> specified and outputs an analysis of it to a file or device as specified.

A. Gibbons

EMAS 2900 UTILITY NOTE

No: 20
Date: 30/3/81

SHOWTAPE

This is an interactive tape examination program, driven by commands similar to ECCE SHOW. The position of each byte of data on the tape is defined by its byte, block, and file displacement from the start of the tape. Commands exist to move a notional cursor to any byte, and to output one or more bytes onwards from the current cursor position in octal, hexadecimal, character (optionally translated), or in a combined form of hexadecimal and character side by side.

Primitive commands consist of an operation, generally specifying a cursor-moving operation or an output operation, followed by a repetition count. For a cursor-moving operation the repetition count specifies the number of bytes, blocks or files by which the cursor is to be moved; for an output operation the repetition count specifies the number of bytes onward from the current position to be output.

The cursor-moving primitive commands are:

Un - move the cursor to byte 1, block 1 of the nth file Up the tape from the current file.(Up is towards the end of the tape.)

Dn - move the cursor to byte 1, block 1 of the nth file Down the tape from the current file.

Fn - move the cursor to byte 1 of the nth block Forward from the current block, within the current file. (Forward is towards the end of the file.)

Bn - move the cursor to byte 1 of the nth block Backwards from the current block, within the current file.

Rn - move the cursor to the nth byte Right from the current byte, within the current block. (Right is towards the end of the block.)

Ln - move the cursor to the nth byte Left from the current byte, within the current block.

The output primitives cause the output of n bytes of the current block starting with the current byte:

On - output in Octal.

Hn - output in Hexadecimal.

Pn - output as Printing characters.

Cn - output in Combined hexadecimal and character side by side.

Clearly such primitives can fail; for example if the specified cursor destination does not exist, or if there are insufficient bytes remaining in the current block to output.

If any output command or the cursor-moving command R fails, the end of block failure message

```
**EOB** AFTER BYTE n
```

is produced. (In the case of an output command the output up to the failure is produced.) If the cursor-moving command L fails, the start of block failure message is simply

```
**SOB**
```

since it must have failed at byte 1. Similarly, the F and B commands produce the file failure messages

```
**EOF** AFTER BLOCK n
```

and

```
**SOF**
```

respectively, and the U and D commands produce the tape failure messages

```
**EOT** AFTER FILE n
```

and

```
**SOT**
```

respectively.

The repetition count for a primitive command can be of two forms:

a) explicit - a positive integer, meaning do it that many times (an explicit repetition count of 1 need not be typed)

b) indefinite - 0 or ?, meaning do it until EOT, SOT, EOF, etc., and produce the appropriate failure message.

Two types of output have already been mentioned - that produced by the O, H, P and C commands, and failure messages. A third type of output is produced when an explicit U, D, F or B command succeeds.

Whenever an explicit U or D command succeeds, the file descriptor for the newly current file is output in the form

```
FILE n
```

Whenever an explicit F or B command succeeds, the block descriptor for the newly current block is output in the form

```
BLOCK n LENGTH=m BYTES
```

Multiple primitive commands can be input on the same line, forming a compound command whose elements are executed from left to right. Additionally, parentheses may be used to define compound commands which, exactly like primitive commands, are followed by an explicit or indefinite repetition count. (An unparenthesised compound command of one or more primitive commands behaves exactly as if it had outer parentheses and a repetition count of 1, and should be so considered in the description of termination and failure which follows.)

An explicit primitive command terminates when its repetition count is exhausted. It terminates and fails at EOT, SOT, EOF, etc. An indefinite primitive command terminates only at EOT, SOT, EOF, etc. It never fails (although it produces a failure message).

An explicit compound command terminates when its repetition count is exhausted (i.e. all its elements have been executed from left to right without failure n times). It terminates and fails when any element of it fails. An indefinite compound command terminates only when an element of it fails. The compound command itself never fails.
(N.B. in view of which, care should be taken to ensure that an indefinite compound command contains a subsequence whose repetition will ultimately give a failure, otherwise the program will loop indefinitely. In this context note that the output commands do not move the cursor.)

The primitive command \ can be used to cancel any failure in the immediately preceding primitive or compound command.

Some examples follow to clarify these points. At the start of a session the cursor is positioned at byte 1, block 1, file 1 and the file descriptor and first block descriptor are output. From this position the command.....produces the output

U?.....EOT failure message only; i.e. the number if files.

(U1)?.....file descriptors for files 2 onwards to EOT.

F?.....EOF failure message for file 1; i.e. the number of blocks.

(F?U)?.....EOF for file 1, then file descriptors and EOF for all files to EOT.

((F1)?U)?.....block descriptors for blocks 2 onward to EOF for file 1, then file and block descriptors for all blocks in all files from 2 onward to EOT.

(H50\F1)?.....hex of first 50 bytes in block 1 then block descriptors and hex 50 for all blocks from 2 onward to EOF; note the use of the \ to cancel failure in the event of a block having fewer than 50 bytes, which would otherwise terminate the sequence.

(F1B1)?.....loops indefinitely without output.

(P10)?.....the first 10 characters of block 1 indefinitely.

(F?)?.....EOF for file 1 indefinitely.

Now to the question of the cursor position after failure. With three exceptions there is always a defined current file, block and byte. As mentioned above, at the start of a session the cursor is positioned at file 1, block 1, byte 1 unless

a) the first file contains no data blocks (i.e. a tape mark at the start of tape) when the empty file condition is raised. The file descriptor and the empty file failure message is output. Thereafter, the commands described so far except U and D produce an appropriate failure message and terminate. This condition can only be raised at file 1. Elsewhere it would correspond to two adjacent tape marks which signals EOT. The EOT condition is raised when a U command is executed and the current file is the last on the tape. Thereafter, the block and byte positions are undefined, commands F,B,L,R and the output commands give an appropriate failure message and terminate and U gives the EOT failure message and terminates. Only D of the commands described so far is effective,

or b) the first block of the first file is a bad block (i.e. one which cannot be successfully read from the tape because of parity errors), when the bad block condition is raised; this condition is also raised when a U,D,F or B command is executed and the newly current block is bad. A warning message is output in the form

```
** BAD BLOCK **
```

The block descriptor takes the form

```
BLOCK n   ** BAD BLOCK **
```

and the current byte becomes undefined. Thereafter the output and L and R commands give an appropriate failure message and terminate.

Apart from these three cases, the cursor position after failure is as follows:

command	file	block	byte
D	1	1	1
F	current	last	1
B	current	1	1
R	current	current	last
L	current	current	1
output	unchanged		

Output from the execution of each O, H, P or C command begins on a new line. Within one such command, newlines are inserted if required on the basis of the page width which by default is 132. This width can be reset at any time using the W command:

Wn - set the page width to n; n<12 or n=? produces a failure message and fails.

The output produced by the O and H commands is always an exact representation of the bit pattern on the tape. The character output produced by P and as part of C is a translated form of the bit pattern.

The translation is controlled by the current translation mode. If the mode=0 (the default) the tape bytes are assumed to be ISO characters and the bottom 7 bits of each byte are used, non-printing characters being translated to space and newline to ^. If the mode=1 the tape bytes are assumed to be EBCDIC characters and are translated to their ISO equivalents, non-printing to space and newline to ^. The translation mode can be changed at any time using the M command:

Mn - set the translation mode=n; n>1 or n=? produces a failure message and fails.

After some interactive movements within the files on a tape, it may be convenient to have the current cursor position output explicitly. This is done using the T(Tell) command:

Tn - output the current cursor position (current file descriptor, block descriptor and byte number); the repetition count is discarded.

The analysis is stopped and the tape released using the S command:

Sn - end the analysis, rewind and release the tape and stop.

Any multi-character console interrupt will cause the current command line to be abandonned, after which it would be advisable to use the T command to locate the cursor.

.....

The program currently resides in ERCC15.SHOWTAPEY on the 2980 and is run by

Command:SHOWTAPE(<tape id>, <in>, <out>)

The tape identifier is required. <out> is where the output is sent and defaults to .OUT. <in> is the source of the command input and defaults to .IN. However, since it can be a file, an arbitrary number of tailor-made analyses can be obtained simply by inserting the appropriate command string in a file (suitably pretested for indefinite looping, especially if the output is not to the console). As a final example, one likely candidate for inclusion in such a set of analyses is the command string

M1((C?F1)?U1)?S

which produces - the file descriptor, the block descriptor, a side by side hex/character dump of all the bytes in the block, for all the blocks, for all the files in an EBCDIC coded tape, the output being suitable for a page width of 132.

.....

Appendix A - Primitive commands

Un - move the cursor up n files.

Dn - move the cursor down n files.

Fn - move the cursor forward n blocks.

Bn - move the cursor backward n blocks.

Rn - move the cursor right n bytes.

Ln - move the cursor left n bytes.

On - output n bytes in octal.

Hn - ouput n bytes in hexadecimal.

Pn - output n bytes as printing characters.

Cn - ouput n bytes in combined hex/character side by side.

Wn - set the page width to n.

Mn - set the translation mode to n.

Tn - tell the current cursor position.

Sn - rewind, release the tape and stop,

\ - cancel failure in immediately preceding command.

.....

C.D. McArthur

EMAS 2900 UTILITY NOTE

No: 21
Date: 30/3/81

BECCE: Binary ECCE (Version 2)

BECCE is an editor available on EMAS at ERCC. It provides facilities for examining and modifying an Edinburgh Subsystem file of any type, presenting it to the user as a series of bytes without regard to any internal structure which it might have (e.g. object file, fixed format data file, partitioned file).

It is based on the text editor ECCE - in fact it is a version of ECCE, with some facilities removed and others added. The use of ECCE is described in detail in the EMAS 2900 User's Guide (2nd Edition), and the reader of this description of BECCE is assumed to be conversant with the facilities provided by ECCE and the way in which they are presented.

Summary of approach

ECCE operates in terms of a 'current line' of the file being edited, and of a 'current position' within the current line. Text specified (for various purposes) by the user is given as a string of characters. ECCE's output consists of the current line as a string of characters, with the current position indicated.

A "line" in the file is delimited by newline characters. This brings us to the first problem in applying the ECCE approach to "binary" files - files which are to be regarded simply as a series of bytes: there is no certainty that the newline character will appear with sufficient regularity (if at all) to enable the file to be split up into convenient 'lines'. The second problem is that the bytes in the file being edited need not correspond to printing characters, and some alternative means of representing them, for input and output, must therefore be provided.

The approach adopted in BECCE to solve the 'line' problem is as follows: on entering BECCE the user is asked to give a record length in bytes (default 20); he is then asked to give a 'separator' byte, expressed as a decimal value. If he does not give a value, BECCE repeatedly scans the file to be edited until a byte pattern, starting with 255 decimal and working downwards, is found which does not occur in the file. If one cannot be found, BECCE selects an infrequently occurring byte pattern and uses that. It also gives a warning that this has been done.

BECCE then copies the input file, inserting the separator byte at intervals corresponding to the specified record length. Any occurrence in the input file of the separator byte is represented in the copy by two such bytes. The editor proper is then entered with the separator byte doing duty as the 'newline' character. The file being edited is thus made up of a series of 'lines' of identical length (the specified record length), apart from occurrences of two separator bytes together (as explained above); these would appear as blank lines in the file being edited. As stated above, no account is taken of any internal structure which the file might have, and it is up to the user to ensure that any such structure is preserved.

At the end of the edit session, BECCE removes all single occurrences of the separator byte on copying the workfile to the output file, and (if no unique byte was found at the start of the editing) each occurrence of two separator

bytes together is replaced by a single separator byte. Thus if the user wishes to insert the separator byte at some point in the file he can do this by inserting a blank line (e.g. by use of R*B or L*B or BB) during the editing. Note that if the edited file is not required, it is quicker to exit from BECCE by means of 'INT:A' rather than '%C'.

The file header is not made available for editing. For data files, the "no of records" figure in the header is adjusted to the appropriate value at the end of the edit session, if the file format is F; if the format is V (or Unstructured) no such adjustment is made.

Byte formats

As noted above, the contents of a binary file cannot in general be represented by character strings. BECCE allows the user to indicate how his input strings are to be specified, and how the output 'lines' are to be printed. It does this by means of two new '%' commands:

1) >%I=<letter>

The %I command specifies the input format of strings in commands to BECCE. <letter> can be one of

H or h	for hexadecimal
D or d	for decimal
O or o	for octal
C or c	for character

With each letter there corresponds a 'minimum number of symbols' to represent any byte. These are as follows:

H	2	(e.g. A4)
D	3	(e.g. 194)
O	3	(e.g. 176)
C	1	(e.g. z)

Examples of input strings, with format H:

f/A31f24/ s/7B C 34/

There are no spaces in the 'f' string and so the 'minimum number of symbols' value, in this case 2, is used to split up the hexadecimal digits. Thus f/A31f24/ corresponds to f/A3 1F 24/. If space characters are present they delimit the specification of each byte. Note in the 's' string above that it is not necessary to specify two digits if one is sufficient, so long as spaces are used. Note also that f/A31F2/ would be interpreted as f/A3 1F 02/.

Further examples of input strings, this time with input format D:

f/140034128/ s/45 2 254/
i/P37/ v/64qe/

The first line above follows on from the hexadecimal examples. The second line includes characters which are not digits in the range 0-9. This is allowed: if a character is encountered which is not a digit of the specified input format then it is treated as a 'C' format byte specification in itself. Effectively one can use the 'C' input format whenever convenient. Note that f/1q/ is acceptable: it is treated as f/001 q/. These remarks apply to all the numerical input formats.

In the case of the 'O' and 'D' formats, the 'minimum number of symbols' is 3. However, if the value resulting from taking three adjacent digits is greater than 255 decimal, then the first two digits alone are assumed to define a byte. For example, /15151/ specifies bytes 151 and 51, but /51515/ specifies bytes 51, 51 and 5. To avoid confusion, it is better to use spaces or leading zeroes rather than exploit this rule.

In the case of the 'H' input format, the letters 'A' to 'F' (and 'a' to 'f') are treated as hexadecimal digits 'A' to 'F'.

It follows from the above that every possible input string has a meaning.

The default input format is H.

Finally, note that %L is the default, not %U. It is recommended that %U be not used, as the effect of case conversion can be confusing when a numerical input format is in use.

2) >%O=<letter>[<1-9>][C]

(The parts in square brackets are optional; the square brackets themselves are not typed)

This specifies the format of the output lines produced by BECCE. The letters permitted are the same as for %I. The optional digit following the letter specifies how many character positions are to be allotted. If this is not specified, the 'minimum no. of symbols' plus 1 is used; this results in each byte being output preceded by a single space. If a value less than the 'minimum number of symbols' is given then it is ignored and the default is taken.

The letter 'C' (or 'c') can be appended to the %O command. It causes each line output by BECCE to be repeated in 'C' format.

If the output format is C then bytes which do not correspond to printable characters are represented by '_'.

The default output format is H3.

Changing the size of the file

The use of some ECCE commands change the size of the file, e.g. e, k, i/text/. However, since there are applications where it is intended not to change the file size but merely to replace bytes by the same number of new bytes, a warning can be generated when the current position is moved off a line whose length is not a factor of the specified record length; the text "Line?" is printed.

To switch this warning on: >%?=1

To suppress it: >%?=0

ECCE facilities not in BECCE

The following ECCE facilities are not available in BECCE:

* Secondary input (but see the next section)
* Secondary output
* SHOW command

Marking and using text already in the file

BECCE enables the user to mark a string of bytes in the file for subsequent use with any of the text location and manipulation commands, viz. D, F, I, S, T, U, V.

The symbol '^' can be used in a command line. It causes the current position to be noted. Subject to certain conditions detailed below, a second use of '^' on the SAME file line causes a string of bytes to be defined, i.e. the bytes between the two positions noted.

If any of the commands D, F, I, S, T, U, V is subsequently used in a command line followed by '@' rather than the usual string specification, e.g. I@ instead of I/Fred/, then the string most recently defined via '^' is implied.

```
Example:   2A 73 2B^ 64 01 FF     (current line)
           >L ^ R* ^ B I@ M- P2   (input to ECCE - spaced out for clarity)
           2A 73 2B 64 01 FF
           2B 64 01 FF           (this line was created by B and I@)
```

Notes

* The file being edited must not be altered between the two uses of '^'. If it is, the earlier '^' is ignored.

* A pair of '^'s can define a null string - a subsequent use of @ would then always succeed. Initially the @ string is null.

* A string can be defined 'the wrong way round'; i.e. the first '^' can point to a position beyond the second '^' position.

* The only failure with '^' occurs when the defined string is longer than 100 bytes.

* The @ string can be redefined as often as desired.

Access

Access to BECCE is via directory CONLIB.GENERAL:

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)
```

Thereafter BECCE is invoked with file specification identical to that of ECCE:

```
Command:BECCE(input,output)
    or Command:BECCE(input)
    or Command:BECCE(,output)
```

(Lack of) Support

BECCE is not supported by the ERCC Advisory Service. While I shall try to fix errors, I cannot undertake to do so as a matter of urgency. I am not willing to change BECCE, but am prepared to give a copy of the source to anyone who would like to.

John M. Murison

Volumes

EMAS 2900: Volumes Note 1

The following new facilities are available in Volumes version 17A onwards. They are described in detail in the attached note.

* Exclusive as well as inclusive user lists for BACKUP, ARCHIVE, EXPORT, etc. (see Section 3.2)

* SUSPEND BACKUP or ARCHIVE or EXPORT (see Section 3.2)

* File system re-creation (see Section 6)

  (File system re-creation is only partially operational as yet. A secure tape is being written nightly, and RECREATE will go through all the motions described herein. However, backup entries are not yet being inserted in users' #ARCHs, so the processing of the latter yields an empty control file; thus no REPLACE requests are generated.)

C.D. McArthur

EMAS 2900 VOLUMES NOTE

The Facilities Provided by the Volumes Process

No: 1
Date: 16/3/79

[This note supersedes all previous Volumes Notes]

Contents

1. Introduction

2. VOLUMS services initiated by users' Directors

3. VOLUMS services initiated by operator

   3.1 Management of private devices and volumes

   3.2 Management of backup/archive store

   3.3 Monitoring

4. Management of backup, archive, export and secure tape lists

5. File system re-creation

6. EMAS 2900 standard tape format

---

1. Introduction

The Volumes process VOLUMS is privileged (ACR 5) and is started as process number 2. It resides on a fixed site on System discs, although it can be started as any file in the file system.

VOLUMS has a file index in each file system, in which it keeps lists of spool requests, tape lists, lists of files awaiting backup, archive, or export, and log files. (Process 1, DIRECT, also uses these indices, having none of its own).

The services provided by VOLUMS include the management of private devices and volumes and the maintenance of the backup/archive store.

---

2. VOLUMS Services initiated by users' Directors

The activity numbers of these services lie in the range 64-127. Those currently defined are as follows:

Activity 67: Notification of a user process stopping.

Request: integer DEST, SRCE, P1, P2, P3, P4, P5, P6

Action: Any volume requests outstanding for this user are cancelled, and any volumes allocated to him released.

Reply: None

Activity 68: Request for the allocation of a volume.

Request: integer DEST, SRCE, IDENT, TYPE, PERM, string (7) DSN, c
integer P6

TYPE = 4 - tape (currently the only type defined)

PERM = 1 - read
= 2 - write
= 3 - read/write

DSN = identifier of required volume

Action: If the volume is not currently loaded the operators are requested to load it with the appropriate permission; if it is currently loaded with the wrong permission they are requested to unload it and reload it with the required permission. These requests may be refused by the operators. If (or when) the volume is correctly loaded, it is allocated to the requesting user.

Reply: integer DEST, SRCE, IDENT, FLAG, SNO, string(7) DSN, c
integer P6

FLAG = 0 - success
# 0 - failure

SNO = service number of successfully allocated volume

Activity 69: Request to release a previously allocated volume.

```
Request:    integer DEST, SRCE, IDENT, TYPE, SNO, string (7) DSN, c
            integer P6

            TYPE = 4  - tape
            SNO  = service number of volume
            DSN  = identifer of volume

Action:     The volume is de-allocated from the requesting user, and
            either allocated to another user or unloaded.

Reply:      integer DEST, SRCE, IDENT, FLAG, P3, P4, P5, P6

            FLAG = 0 - success
                 # 0 - failure
```

Note: This activity is automatically invoked by a call on activity 67 ("user process stopping").

Activity 70: Enter a spool request of type "restore" into request list.

```
Request:    integer DEST, SRCE, string (6) TAPENAME, USERNAME, c
            byteinteger FSYS, TYPE, integer CHAPTER, IDENT

            TAPENAME = identifier of source volume
            USERNAME = identifier of requesting user
            FSYS     = file system number of requesting user
            TYPE     = 0 - "restore" spool (see below)
            CHAPTER  = chapter number of file on source volume

Action:     The request is entered in the request list on fsys FSYS, and
            if necessary an allocation request (see activity 68) is made
            in VOLUMS' name.

Reply:      integer DEST, SRCE, FLAG, P2, P3, P4, P5, IDENT

            FLAG = 0 - success  (N.B. means only that request entered)
                 # 0 - failure
```

Notes:

* USERNAME need not be the owner name found in the chapter header when the spool request is subsequently carried out. However, the file is always restored to the owner.

* The file is created with its access permissions, except OWNP, set from the current permissions in the owner's archive index. OWNP is set from the chapter header (i.e. the OWNP in effect at the time the file was written to tape). The status bytes are otherwise not reset.

* The requested tape must have been written in the standard format (see Section 6 below).

3. VOLUMS services initiated by operator

These services are requested by pressing the "COMMAND" key on the Operator's console and typing

V/<command>

followed by the "ENTER" key.

When the requested service does not result in a visible display, there will be an immediate response, as follows:

DONE - The service has been successfully completed.

OK - The command has been recognised and the requested service started. Subsequent conditions may cause the service to fail, but any such failures will be reported explicitly.

END service - The service has ended, not necessarily successfully: there may have been error conditions which will have been explicitly reported.

The services fall roughly into three groups:

1. management of private devices and volumes (Subsection 3.1)
2. management of the backup/archive store (Subsection 3.2)
3. monitoring (Subsection 3.3)

3.1 Management of private devices and volumes

<dev>	:= device mnemonic;	e.g. M50, M61
<devgroup>	:= <dev> ! <dev> truncated on the right	e.g. M50, M5, M
<ident>	:= volume identifier;	e.g. NRA024
<offon>	:= OFF ! ON	

Each of the commands in this group is entered on a single line.

<devgroup>?
Display, in an abbreviated form, the current status of the device(s) specified, on the main operator console; e.g.

V/M50? displays status of M50
V/M5? displays status of M50, M51, etc.

DEVS Display comprehensive status of all devices, on the auxiliary display.

<dev>=<ident>
The unlabelled volume on drive <dev> is to be known by the identifier <ident>; e.g. V/M51=NT4012

NO <ident>
The volume <ident> is not available, and outstanding requests for it are to be refused.

REQ List all outstanding volume requests on the auxiliary display.

RETAIN <dev>
RETAIN <ident>
RETAIN
The device <dev> or the device on which volume <ident> is currently mounted is to remain loaded even when the volume is not in use. If neither <dev> nor <ident> is given, all devices are "retained".

UNLOAD <dev>
UNLOAD <ident>
UNLOAD
The converse of RETAIN; this is the default.

TAPES? Display the Supervisor state of all tape drives, on the main console.

LABEL <dev> <ident>
Write the volume identifier <ident> on the volume currently mounted on device <dev>.

STIPL <dev> <ident>
Write an "S" series IPL label <ident> on the volume mounted on device <dev>.

TIPL <dev> <ident>
Write a "P" series IPL label <ident> on the volume currently mounted on device <dev>.

MODE <dev>=<offon>
Tell the Supervisor to release (OFF) or reclaim (ON) the device <dev>.

3.2 Management of the backup/archive store

```
<backup type>        := ALL!CHANGED!WEEKLY!DAILY
                        (WEEKLY= all cherished files,
                         DAILY = all changed cherished files)

<archive type>       := DESTROY!TO TAPE

<export type>        := ALL!CHANGED!WEEKLY!DAILY!SELECT

<index backup type>  := NEW!APPEND

<index restore type>:= ALL!null

<tape type>          := BTAPE!ATAPE!ETAPE!STAPE

<fsys>               := file system number (decimal)

<any or all fsys>    := <fsys>!-1
                        (-1 means that the actual file system number is
                         unknown, or all file systems are to be included.
                         If this is the last item on the line it may be
                         omitted.)

<single user>        := user name with all 6 characters defined

<set of users>       := user name with one or more characters undefined
                         by '?'

<user group>         := <single user>!<set of users>

<inclusive user list>  := (<user group><rest of user list>)

<exclusive user list>  := )<user group><rest of user list>(

<user list>          := <inclusive user list>!<exclusive user list>

<rest of user list> := ,<user group> <rest of user list>!null

<userspec>           := <user group>!<user list>!null
                         . (null = <user group> of ??????)

<first>:=<last>:=<page>:=<pos>:= decimal number
```

If an inclusive user list is specified, all the users on the nominated file systems(s) who match anything in the list take part in the operation. If an exclusive user list is specified, all the users on the nominated file system(s) who do not match anything in the list take part.

Several of the commands in this group require to be entered on multiple lines; for the second and subsequent lines an appropriate prompt is output, and the response should be given without the preceding "V/".

BACKUP Initiate a backup sequence. The prompt "GIVE BACKUP SPEC" is output and a line of the form

<backup type> <any or all fsys> <userspec>

should be entered; <userspec> can be continued over multiple lines, when the prompt becomes "CONTINUE INCLUSIVE LIST" or "CONTINUE EXCLUSIVE LIST", as appropriate. Individual list items may not be split across a line boundary. The nominated files are written to tape(s) taken from the BTAPE list, a listing of tape contents is produced, the write bits for the files are cleared, the #ARCHs are updated and a secure tape is written (see Section 5).

ARCHIVE Initiate an archive sequence. The prompt "GIVE ARCHIVE SPEC" is output and a line of the form

<archive type> <any or all fsys> <userspec>

should be entered; subsequent lines are as for BACKUP. Files eligible for archive are those which have been unused for some period (see MARK UNUSED, below) or those which have been explicitly marked for archiving by the user. If <archive type> is "DESTROY", eligible files which are not cherished are destroyed and a listing is produced. If <archive type> is "TO TAPE", eligible files which are cherished are written to tapes taken from the ATAPE list, the #ARCHs are updated, a listing of tape contents is produced and the files are destroyed.

EXPORT Initiate an export sequence. The prompt "GIVE EXPORT SPEC" is output and a line of the form

<export type> <any or all fsys> <userspec>

should be entered; subsequent lines are as for BACKUP. If <export type> is SELECT a prompt of the form

SELECT user.filename?

is output for each file and each specified user, to which the response "Y" (yes), or "N" or null (no), should be given to include or exclude the file. The nominated files are written to tape(s) taken from the ETAPE list and a listing of tape contents is produced. (Note that the SELECT file confirmations are done using DOUT, and so other VOLUMS activities are suspended until the selection is complete. Since it is possible by accident or design to specify all users, a very large number of responses could be required. Two "get-outs" are available: the response "NEXT" to a file implies "N" to that file and to the remaining files for the current user. The response "END" implies "N" to that file and to the remaining files for the current user, and to all files for the remaining users.)

RESUME BACKUP
RESUME ARCHIVE
RESUME EXPORT
RESUME RECREATE
Continue a backup, archive or export sequence which was interrupted. See Section 5 for RESUME RECREATE.

RECORD BACKUP
RECORD ARCHIVE
RECORD EXPORT
: If BACKUP, or ARCHIVE TO TAPE, or EXPORT, proceed to that part of the sequence (described above) immediately following the writing of tapes; if ARCHIVE DESTROY, proceed to destruction.

SUSPEND BACKUP
SUSPEND ARCHIVE
SUSPEND EXPORT
: Terminate the sequence as soon as possible consistent with a resumable state; the current writing or reverse checking of a file is allowed to complete. Tape(s) which are currently being written or reverse checked have a "CHECK FAILS" forced; tape(s) which have been successfully written and checked remain valid. If load requests are outstanding, it is necessary to follow the SUSPEND by a NO <tape> in order to kick the appropriate activity.

  The control files are retained and the sequence can be RESUMEd. If, instead, the sequence is restarted from scratch the retained control files are discarded.

MARK UNUSED
: Mark files which have not been used since the last MARK UNUSED was done. When a file accumulates a number of such unused marks it becomes eligible for archive.

INDEX BACKUP
: Initiate an index backup. The prompt "GIVE INDEX BACKUP SPEC" is output and a line of the form

    <index backup type> <any or all fsys> <userspec>

  should be entered; subsequent lines are as for BACKUP. The non-file index information for the specified file system(s) and users is collected in a file called INDEXBACKUP (one per fsys) in VOLUMS' index. If <index backup type> is NEW, any existing file is first destroyed. If <index backup type> is APPEND there must be an existing file to which the information is appended. APPEND should be used with caution since, if a user's information appears more than once in a file, only the earliest can be retrieved by INDEX RESTORE. The INDEX BACKUP performed automatically as part of the secure operation (see Section 5) is always of type NEW.

INDEX RESTORE
: Initiate an index restoration. The prompt "GIVE INDEX RESTORE SPEC" is output and a line of the form

    <index restore type> <any or all fsys> <userspec>

  should be entered; subsequent lines are as for BACKUP. The non-file index information contained in the file INDEXBACKUP (found in the VOLUMS index on the specified file system(s)) is used to re-accredit the specified users on that file system. If such a user is currently accredited anywhere else he is not re-accredited.

The file found on a file system need not have originated there. Thus users can be re-accredited on a file system other than their original.

RELOAD <ident> <first> <last>
Spool the files in chapters <first> to <last> inclusive on tape <ident> into the on-line file system. If neither <first> nor <last> is given, all chapters on the tape are implied; if <last> is not given, the single chapter <first> is implied.

The files are created with access permissions, arch byte, codes byte, cct and ssbyte set from the tape file header (i.e. as they were at the time the tape was written). Except for VOLUMS and SPOOLR files, which are re-created on the file system where they originated, all files are created on the file system where the owner is currently accredited.

TRANSFER <ident> <first> <last>
As RELOAD (above), except that existing files of the same names in the destination file system are first destroyed, and the status bytes are not reset.

REPLACE <ident> <first> <last>
As RELOAD (above), except that there is a prompt for the destination file system. The value given is used instead of any which might be in the tape file header.

If a file's owner is not currently accredited on the specified file system the file is not spooled.

RECREATE <fsys>
Initiate a re-creation with <fsys> as the destination file system. The prompt "GIVE TAPE SRCE USERSPEC" is output and a line of the form

<ident> <fsys> <userspec>

should be entered; subsequent lines are as for BACKUP. <ident> is the identifier of the secure tape to be used and <fsys> is the source file system on that tape. See Section 5 for a detailed description of RECREATE.

DISCARD BACKUPS
For all file systems and all users, discard #ARCH backup entries which have expired. See Section 5 for a detailed description.

REQLIST <page>
List page <page> of outstanding spool requests, on the auxiliary display.

BADREQ <ident> <first> <last>
Delete, from the request lists, the spool requests for tape <ident>, chapters <first> to <last> inclusive.

NEW <tape type>LIST

Create a new tape list of the specified type on the current IPL file system (see Section 4). The list is created empty.

ADD <tape type> <ident> <pos>

Add the identifier <ident> to the specified tape list at position <pos>, where 1 <= <pos> <= current length of list+1.

If <pos> is not specified the end of the list is assumed.

REMOVE <tape type> <ident>

Remove the identifier <ident> from the specified tape list.

<tape type>S

Display the specified tape list on the auxiliary display.

3.3 Monitoring

Each of the commands in this group is entered on a single line.

PRINT Print the current VOLUMS log on the line printer and initialise a new log file.

MONON Switch on monitoring.

MONOFF Switch off monitoring.

STOP Stop.

---

4. Management of the BACKUP, ARCHIVE, EXPORT and SECURE tape lists

<tape type> := BTAPE ! ATAPE ! ETAPE ! STAPE

The four tape lists reside in files VOLUMS.<tape type>LIST. They are created and initialised on the current IPL file system (or, if the file already exists on that file system, reinitialised) by the NEW command; see Subsection 3.2.

Tape identifiers are added to a list by ADD and removed from it by REMOVE.

Note that when adding a tape identifier to a list, or retrieving one for use, the available file systems are searched from 0 to find a file with the appropriate name. It is thus important that the first one found is the current one, and in practice there should only be one list file of each type across all file systems. If a new list file is created on a file system other than that which contains the current one, the latter should be subsequently deleted by MANAGR. (VOLUMS does not attempt this since the file system may be off-line.)

The current contents of a tape list are displayed on the auxiliary display by the <tape type>S command.

The identifiers of tapes to be written are taken from the front of the lists. In the case of backup or secure tapes they are immediately replaced at the back; i.e. these lists are cyclic. These two lists also contain, for each tape, the date when it was last written (see Section 5).

The ETAPE list used by EXPORT is not managed, and the ETAPES and ADD commands should be used prior to the export to ensure that the required tapes are at the front of the list. It would be good practice to leave this list as it was found, if necessary using the REMOVE command.

---

5. File System Re-creation

This section includes descriptions of archive indices, secure tapes, the backup tape cycle and backup tape list (BTAPE list), and index backup and restoration insofar as they contribute to file system re-creation.

Each user has an archive index (#ARCH) which contains, in addition to archive entries, entries for all the user's files on backup. Backup tapes are used cyclically, and so these backup entries have only a temporary validity. Since a #ARCH containing references to a particular backup tape may be unavailable (e.g. off-line) when that tape is about to be rewritten, it is not possible (nor would it be efficient) to search for and discard such backup entries at that time. Instead, a backup entry is validated by comparing its date with the date when the tape was last mounted with a write ring (this is held in the BTAPE list). If they are not equal the entry is no longer valid. Discarding such expired entries is thus a housekeeping operation which can be performed at any time using the command DISCARD BACKUPS.

Thus, given

1) the BTAPE list,
2) the set of users' #ARCHs, and
3) the backup tape cycle

it is possible to re-create automatically all users or a group of users, on one or more file systems. This has been implemented as follows.

Following the daily backup, when all available #ARCHs have been updated, an INDEX BACKUP(NEW) is automatically performed which collects users' non-file index information in a file called INDEXBACKUP (one per fsys) in VOLUMS' index. Then a further tape, called a "secure tape", is written containing the BTAPE list, and for each file system this index backup file followed by the users' #ARCHs in the same order as the users' entries occur in the index backup file.

To re-create all users or a group of users on a destination file system, it is only necessary to nominate the secure tape to be used and the source file system on that tape. Thus:

```
V/RECREATE <destination fsys>
```

after which the prompt "GIVE TAPE SRCE USERSPEC" is given.

A search is first made for the BTAPE list on all on-line file systems. If it is not located it will be retrieved from the secure tape during the following procedure. A scan is made of the secure tape to locate the index backup file for the source file system. This file is REPLACEd on the destination file system and used to do an INDEX RESTORE for the specified users, which re-accredits them and puts back their non-file index information (password, delivery information, etc). Then the #ARCHs for the same users are located on the tape (immediately following the position of the index backup file) and REPLACEd on the destination file system. Any missing #ARCHs, unavailable at the time the tape was written, are detected and reported at this time.

When all the available #ARCHs have been REPLACEd, the backup entries in them for the specified users are processed to obtain the entries for the most recent versions of all their files on backup, and ordered to be

retrieved in a single pass through each required backup tape. Standard REPLACE requests for all the necessary tape chapters are then generated. (Only a single request is generated for adjacent chapters, but there could still be a very large number of requests. The request lists on all on-line file systems are used if necessary, but if all these fill up the generation is suspended and can be resumed when space becomes available by the command RESUME RECREATE.)

Since only the most recent versions of files are requested, the order of mounting the backup tapes is not important. They are requested in ascending order of the tape sequence number, and the REPLACEs may proceed in parallel, using as many tape drives as are available.

The RECREATE operation is officially complete when all the REPLACE requests have been generated, since these are standard requests and will be preserved across a crash. The system could be opened to the users at this point since they are accredited, their files gradually returning as the REPLACE requests are actioned.

If more than one file system is to be re-created, the next one can be started at this point. The REPLACE requests generated for it will join those already in the request lists and the normal mechanism for selecting the next request to be actioned will effectively merge them.

Note that the scan of the secure tape is triggered by a single REPLACE request for all chapters on the tape, and the integrity of the scan depends on there being no other requests (e.g. RELOAD, TRANSFER, etc.) for the secure tape while the scan is in progress.

Secure tapes to be written are taken from a small cyclic list called the STAPE list. A listing of the secure tape contents is produced as for backup etc., once it has been successfully written and checked. Writing the secure tape is considered to be part of recording the backup. Thus there is no explicit command to initiate it. If, following a successful backup, the secure tape is not written for some reason, this can be achieved by RECORD BACKUP. (This will note in passing that the BACKUP control files do not exist.)

A full scale RECREATE consists of a sequence of operations which can be initiated in isolation if required, as described below. Notice, however, that it is always safe to do a complete RECREATE, since none of its operations are destructive: if a user is currently accredited anywhere, he will not be reaccredited on the destination file system; existing on-line files will not be overwritten by any coming from backup, since REPLACE is non-destructive. Even if the source or destination file system is specified incorrectly, the worst that could happen is that user-destroyed files might reappear on the destination file system.

If the secure tape identifier is given as "SKIP", RECREATE assumes that the required users are already accredited on the destination file system and have up-to-date #ARCHs. It thus proceeds directly to the processing of backup entries from the latter, bypassing the scan of the secure tape and the index restoration.

Index backup and restoration can be initiated by the commands INDEX BACKUP and INDEX RESTORE. No tape transfers are involved: the non-file index information is gathered into or dispersed from the on-line index backup file in VOLUMS' index.

Such partial re-create operations may be useful in a variety of situations; e.g. when a file system is only partially corrupt, or when HARCHs are missing from the most recent secure tape, or when the secure tape cannot be read successfully (in which case a semi-manual re-create must be done, perhaps involving the next most recent secure tape.)

EMAS 2900 Standard Tape Format

```
Tape format:     80 byte standard VOL1 label
                 TM
                 Chapter 1 (blocksize = epage size)
                 TM
                 Chapter 2
                 TM
                 .
                 .
                 .
                 Chapter n
                 TM
                 TM

Chapter format:  Block 1      header page (see below)
                 Block 2  )
                 .        )
                 .        ) n epages of file
                 .        )
                 Block n+1)
```

Header page format:

```
string(6) TAPENAME, USERNAME, string(15) FILENAME, string(8) c
DATE, TIME, TYPE, byteinteger SPARE0, SPARE1, SPARE2, integer c
CHAPTER, EPAGES, FSYS, PERMS, OWN, EEP, ARCH, CODES, SSBYTE, c
CCT, SPARE3, SPARE4, SPARE5, RECORDS, string(6) OFFERED TO, c
recordarray PERMLIST(1:16) (string(6) USER, byteinteger PERM)
```

TAPENAME	name of the tape this file is on
USERNAME	owner of the file
FILENAME	name of the file
DATE	date when written to tape
TIME	time when written to tape
TYPE	"BACKUP", "ARCHIVE" or "EXPORT", as appropriate
CHAPTER	the chapter number on the tape
EPAGES	the number of epages in the file, not counting the header
FSYS	-1 for all but "VOLUMS" and "SPOOLR", when it is the original file system of the file
PERMS	1 if access permissions are present, else 0
OWNP	owner's access permission at time of writing

EEP	everyone else's permission at time of writing
ARCH	arch byte
CODES	codes byte
SSBYTE	ssbyte
CCT	connect count
RECORDS	number of individual permissions (in permlist)
OFFERED TO	the user to whom the file is on offer, else null
PERMLIST	array of individual permissions

C.D. McArthur