GEC 4000 Series

Volume 2:
PROGRAMMER'S
FACTS BOOK

GEC Computers Limited

OS4000 Facts Book

Volume 2:
PROGRAMMER'S
FACTS BOOK

GEC Computers Limited

CONTENTS

This document contains a summary of information useful to a programmer for OS4000 software. The details enclosed previously formed part of Volume 2: Programmer's Facts Book, 85-62042, which has now been re-structured as three separate books, of which this is one. The remaining two are OS4000 Babbage Language Reference Summary, 85-64730 and Volumne 4: Processor Facts Book, 85-64731.

This document therefore constitutes the (modified) second volume in a set of facts books and should be used in conjunction with the rest of the set.

This series of facts books is intended to supplement, and not replace, the manual set.

TERMINAL-LOGICAL MODE (LT and TTY)

Teletype compatible mode (TTY) is a personality of TF and provides all the logical facilities available to LT plus those asterisked below.

* means the command may only be used with TF/TC.

Command (RY)	Meaning
* @0280	Set terminal control parameters (see the manual System Services, 85-62006). Not recommended when in page-mode. RX Bits 0-7 = Parameter number RX Bits 8-15 = New parameter value
* @0281	Examine terminal control parameters (see the manual System Services, 85-62006) RX Bits 0-7 = Parameter number RX Bits 8-15 = Returned parameter value
* @0FC0	Select personality: RX=0 Default RX=1 Personality TTY RX=2 Personality PHYS RX=3 Page-mode operation where supported RX=4 Reserved RX=5 Personality PAD
* @02EF	Discard current input data. If the command immediately following is not issued in page-mode and is a PUT then output will be forced, regardless of the state of the current input line
@02F0	No CR LF to follow next PUT on this stream
@02F1	No LF to follow next PUT on this stream (has no effect in page mode)
@02F2	No echo on next line input (where terminal supports it)
@02F3	Set Prompt Mode
@02F4	Set Buffer Mode
@02F5	Set new warning character to that in RX (which must be printable)
@02F6	Select this stream as supervisory
@02F7	Set alternative character for warning pairs: RX Bits 0-7 = Basic control letter RX Bits 8-15= Alternative symbol
@02F8	Set prompt length (RX = 0-4). Values greater than 4 set length = 4
@02F9	Lose the contents of the input buffer
@02FA	Control output buffering: RX = 0 Switch off output buffering RX = 1 Switch on output buffering

Command (RY)	Meaning
	RX = 2 Clear buffer and switch off buffering RX = 3 Clear buffer and switch on buffering RX = 4 Clear the output buffer Values greater than 4 are ignored
* @02FB	Program control of muting: RX=0 Mute all other streams RX=1 Mute stream (others unchanged) RX=2 Enable all streams
@02FC	Select the required method of prompting as defined by RX: RX = 0 Set Buffer Mode RX = 1 Set Prompt Mode RX = 2 Set Buffer-Prompt Mode
@02FD	Set Screen Mode as defined by RX: RX=0 Switch off Screen Mode 0<RX<100 Set Screen Mode with length of screen=RX RX= Any other Set Screen Mode with the default length
* @02FE	Select next GET, requires no data returned. This mode automatically implies 'no echo' mode on the GET
@02FF	Set default parameters as specified at system generation. Not recommended when in page-mode.
@0BF0	Divider control RX = 0 Next PUT is text for divider. The text will be truncated if necessary RX = 1 Switch off divider (any divider text will be lost)
@0BF1	Examine window parameters RX = 0 Return in bits 0-7 of RX the MAX of the window and in bits 8-15 of RX the MIN of the window. RX = 1 Return in bit 0 of RX a flag which is set if a divider is present in the window; in bits 1-7 of RX the start line of the window and in bits 8-15 of RX the length of the window
@0BF2	Set window MAX and MIN. The new MAX will be set to the value of bits 0-7 of RX, and the new MIN to the value of bits 8-15 of RX

Physical mode (PHYS) is a personality of TF and provides all the physical facilities available to SPAS plus those asterisked below.

* means the command may only be used with TF/TC.

Command (RY)	Parameter (RX)	Meaning
* @0F40	0	Select one Way Alternate (one Way Control Block) operation. This is the state when the physical stream is opened, and is used for normal operation
	1	Select Two Way Simultaneous (two Way Control Blocks) operation
* @0F41		Disconnect dialled line
* @0F80		Set terminal control parameters (see the manual System Services, 85-62006)
	Bits 0-7 Bits 8-15	Parameter number New Parameter value
* @0F81		Examine terminal control parameters (see the manual System Services, 85-62006)
	Bits 0-7 Bits 8-15	Parameter number Holds parameter value on reply
* @0F82	126	PCC PAD Terminator mask is set to cause data forwarding on all unprintable characters (0-31 and 127)
* @0F83	255	PCC PAD Echo mask is set to suppress echoing of all unprintable characters (0-31 and 127)
* @0FC0	Select Personality	RX=0 Default RX=1 Personality TTY RX=2 Personality PHYS RX=3 Reserved RX=4 Reserved RX=5 Personality PAD
@0FF0	0	Set parity Transparent (default)
	1	Even parity Generated/checked
@0FF1	0	Input not echoed (default)
	1	Input is echoed
@0FF2	0	Input not terminated by specific character (default)
	1	Input terminator specified to B42S (This may require even parity generated/checked to be enabled in order that odd parity string terminators can be detected)
@0FF3	0	Reader Control Off (default)
	1	Reader Control On
		This command is ignored except on Teleprinter-compatible device driver Board 42

Command (RY)	Parameter (RX)	Meaning
@0FF4	0	Terminal interrupt disabled (default)
	1	Terminal interrupt enabled
		This command is ignored by Board 42 (always active)
* @0FF5	0	PCC unsolicited input buffer is used (default)
	1	PCC unsolicited input buffer is discarded on GET
@0FF6	0	Automatic CR LF
	1	No automatic CR LF (default). The terminator is the last byte in the users buffer and there is no new line following
* @0FF7	0	Do not use PAD String Terminator and Echo masks (default)
	1	All GETs to use the PAD String Terminator and Echo masks
@0FF8	LEN	The following PUT is to be treated as a PUT followed by a GET into the same buffer but of length LEN. The GET has a timeout applied, and if it has not been completed when the timeout expires, data management error @850F is returned. The default value of the timeout is 60 seconds. This PUT-GET is intended for use in cursor read and polling applications
@0FF9	Timeout in seconds	Change the timeout to be applied to PUT-GETs (initiated by control @0FF8). A value of zero is treated as no timeout
@0FFF	-	Select default parameters as specified at system generation

Command (RY)	Parameter (RX)	Meaning
@03C3	-	Throw to top of form (or advance paper 2.5 inches on Versatec in roll mode)
@03F0	>0	Set page size to RX (normally 60 lines)
	<=0	Suppress paging
@03F1	-	Overprint the next record output with that following it, i.e. suppress paper feed after the next record. This may be used to generate special characters (e.g. †) and, by outputting all or part of the same record twice, heavy print (not Versatec)
@03F2	n	Skip to channel n (n=0 to 11) (ignored by Versatec/DMP if S/W channels not set)
@03F3	n	Throw n lines (1=<n=<2*page size)
@03F4	n	Throw to line n (1=<n=<page size)
@03F5	X:n	Set S/W channel n at line X (X is most significant byte, n is least significant byte)
@03F6	Not 1	Select S/W channels (default)
	1	Select H/W channels (on Versatec/DMP: unset S/W channels) (initial setting)
@03F7	<1 or >2	No automatic CR or LF (default)
	1	Automatic CR only, i.e. over-printing. This continues until specifically reset (not Versatec)
	2	Automatic CR and LF (initial setting)
		Note: The overprint command (@03F1) takes precedence over this setting
@03F8 (Versatec only)	0	Print following single line
	1	Simultaneous print/plot line follows
@03F9	n	Change open options to n
@03FA	n	Change error options to n
@03FB	n	Set n initial blanks (space in text, null in plot)

Note: If a device is driven by the Multiple Printer Writer (MPW), the following exceptions to the above should be noted:

- @03F6 Only S/W channelling is supported
- @03F8 Is ignored
- @03F9 Is ignored
- @03FA Is ignored

The following control commands are available on streams open in Physical Mode:

Command (RY)	Meaning
@04C4	Perform RX Erases *
@04C8	Perform RX Backspaces (updates block number if necessary)
@04C0	Write RX File Marks (ignored if Ignore Tape Marks is set) *
@0414	Rewind on-line, Block number=0
@040C	Rewind off-line
@04F0	RX=0 set no block numbering RX<>0 set block numbering required (default block numbering only used when header block length = 4)
@04F1	RX=0 Ignore Tape marks (default) RX<>0 Return Tape marks
@04F2	Set header block length = RX (should be 0 or 4). Default=4
@04F3	Set block number for selective overwrite = RX (set -2 for none). Default=-2
@04F4	Number of attempts (=RX) made before returning Data Management error (default=40)
@04F5	Return the number of the next block in RX
@04F6	Set next block number = RX and position ready to READ/WRITE it. If blocks are not numbered, 1 backspace is performed
@04F8	Return Device Status in RX
@04F9	Change tape parity (7 track only) RX=0 set odd parity (default) <>0 set even parity

Any control command given on a stream open in Logical Mode is entered into the current block and may be read back subsequently by opening for input with RX bit 7 set. See Data Management CONTROL routine.

* If end-of-tape is encountered during the execution of these commands, the command is terminated immediately and control is returned to the caller. On return RX will contain the number of operations outstanding, and RA will contain the end-of-tape error code.

Command (RY)	Parameter (RX)	Meaning
@05F0	Open options	Change open options *
@05F1	-	Rewind **
@05F2	n	Backspace n records ***
@05F3	-	Write back block after every update
@05F4		Set offline handling mode (PHYSICAL mode only)
	0	Normal mode (default): offline is not detected; disc requests will be held up until disc comes back online
	1	One-shot error mode: occurrence of offline will cause an error reply, whereupon the stream will revert to mode 0
	2	Full error mode: offline will always cause an error reply

* The permitted changes are from random to sequential, read to write, read to update, write to update and vice versa. Changing to open for reading or updating causes the next record (or block if in Physical Mode) to be read as the first record of the file. Changing to open for writing causes the next record (or block if in Physical Mode) written to be added to the end of the file. When changing open options the non-default error bit should not be set; the options of the original open will be retained, e.g. if a file was originally opened with options @8001, then to change from writing to updating option @0008 should be specified, not @8008.

** This causes the file to be emptied if the file is open for writing and the next record (or block if in Physical Mode) written is the first in the file. If the file is open for reading or updating the next record (or block if in Physical Mode) read is the first record of the file.

*** This causes the file to be backspaced n records (or blocks if in Physical Mode). If n=1 the next record read or written is the record read or written last. Therefore to read a file backwards n must be greater than 1. If n is greater than the number of records (or blocks if open in Physical Mode) that have been read or written so far then the action is as for rewind. If despooling records exist in the file then these are excluded from the count of records to backspace unless the file is opened to return spooled control commands.

Command (RY)	Parameter (RX)	Meaning
@06F0	<=0	Binary Synchronous Communications
	=1	Direct Line Driving
	>=2	Reserved
@06F1		Change Initiate Timeout (Multiple of Receive):
	<=0	For default value
	>0	New value
@06F2		Change Receive Timeout:
	<0	Default value
	=0	None
	>=0	New value
@06F3		Retry attempts:
	<0	Default
	>=0	New value
@06F4		Change terminator:
	0	ETX
	1	ETB
	2	Reserved for ITB
@06F5	0	Transparent mode
	1	Non-transparent mode
@06F6		Commence with header
@06F7		Set Receive Timeout (in milliseconds):
	<0	Default value
	>=0	New value
@06FF	0<tries<4	Precedes a PUT of a number to be dialled
@06FE	Timeout (minutes)	Set Auto Answer mode

For use with synchronous modem half-duplex V24 Controller Board 76.

Initiate Timeout is the timeout value to prevent contention for the line.

Receive Timeout is the maximum acknowledgement delay permitted.

Command (RY)	Meaning
@0A02	Suppress sequence number checking (logical only)
@0A40	Select Coded Mode (physical only)
@0A41	Select Non-Coded Mode (physical only)
@0AF0	Change error options to RX
@0AF1	Change open options to RX

DATA MANAGEMENT

The two co-existing Data Management systems, DM1 and DM2 use different OS4000 mechanisms, but appear similar to the user through the standard routine call interface. DM2 provides a number of additional routines, and also provides facilities for command qualifiers and command identification. Some drivers also provide multiple queuing of commands. However, for all normal use, the two data management techniques provide identical user interfaces, and there is no restriction on the mix of streams used by any one process. The routines, KILL, STATUS, LOCATE and EXCHANGE are only acceptable with DM2. The set of routines available is described below. The following definitions hold for all routines unless otherwise stated:

QUALIFIER

The user's further definition of the operation required. This field is only meaningful for Data Management Type 2, and is then either the number or it is bit orientated, according to requirements of the particular driver concerned. It is set to zero when not used.

E

Set if an error is returned; clear otherwise. The Conditions Register on return from all routines is set on the state of RA so that each return may be followed by a direct test of Conditions.

ERROR CODE

This is only meaningful if E is set and indicates the type of error returned. It is described under DATA MANAGEMENT ERRORS.

ID

The user's identification of the command. It is normally zero, and it should be zero for Data Management Type 1 drivers. For Data Management Type 2 drivers the field is preserved.

OPEN

This routine is used to open a nominated stream for use in input or output modes.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM
RX	OPEN OPTION	OPEN OPTION
RY	ERROR OPTION	ERROR OPTION

STREAM

Stream Number in the range 1=<STREAM=<255

OPEN OPTION

Defines the way that the stream is to be used as follows:

RX bit number	Bit Unset	Bit Set
0	Default error handling	Special error handling
1-5	Reserved	Reserved
6	User supplied buffers	Locate Mode
7	Spooled control commands ignored (when open for input only)	Spooled control commands returned (when open for input only)
8	Logical mode	Physical mode
9	Non-Indexed mode	Indexed mode
10	Reserved	Reserved
11	Normal operation	Paged mode operation
12	Non-update	Update (bit 15 redundant)
13	Sequential	Random
14	Text	Binary
15	Input	Output

Note: Bits 9 and 13 are mutually exclusive.

ERROR OPTION

Defines the user selection for handling errors. If RX bit 0 is 0 (unset) then the error handling selects a default and RY is ignored. If RX bit 0 is 1 (set) then RY defines the error option. See DATA MANAGEMENT ERRORS.

CLOSE

This routine closes the nominated stream after use. The only activity which may follow on this stream is an OPEN or another CLOSE.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM

The return of E and ERROR CODE is as described above.

GET

This routine causes data to be input on a stream provided the stream has been opened for input or updating.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM
RX	AVAILABLE BUFFER LENGTH	LENGTH OF DEFINED DATA
RY	BUFFER ADDRESS	BUFFER ADDRESS

AVAILABLE BUFFER LENGTH

Is the number of bytes available for data in the buffer.

LENGTH OF DEFINED DATA

Specifies how much of the buffer has been meaningfully filled. The remainder of the buffer is undefined. In Logical Mode any terminating characters (e.g. CR and LF) are not included in the length. In Physical Mode some devices return values in RX different from the actual data length. See DEVICE CONTROL COMMANDS AND PARAMETERS.

BUFFER ADDRESS

The address of the first byte of the buffer to be filled. The segment containing the buffer must be in CST0, 1 or 2 and must normally have Read, Write, Transfer and Send access.

The return of E and ERROR CODE is as described above. 8000 is always returned regardless of the error options; this indicates end-of-file. The user should always close the stream following an end-of-file indication so that system compatibility is maintained.

PUT

This routine causes data to be output on a stream provided it has been opened for output or updating. Update mode is not supported by all drivers.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM
RX	LENGTH FOR OUTPUT	LENGTH FOR OUTPUT
RY	BUFFER ADDRESS	BUFFER ADDRESS

LENGTH FOR OUTPUT

Is the number of bytes to be output.

BUFFER ADDRESS

Is the address of the first byte of the buffer to be output, with the same restrictions as for GET.

CONTROL

This routine provides various device specific operations. These commands sent to logical disc or magnetic tape files are recorded in the file and can be re-obtained using GET with a stream opened with the relevant option, i.e. RX bit 7 set on OPEN.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 | QUALIFIER |	E | ERROR CODE
RA(AL)	ID | STREAM	ID | STREAM
RX	CONTROL PARAMETER	CONTROL PARAMETER
RY	CONTROL COMMAND	CONTROL COMMAND

CONTROL PARAMETER

May be used as a request parameter, a return parameter or both.

CONTROL COMMAND

Takes the form @0PQR where P defines the relevant device:

0	Reserved	8	VDU Type 1 *
1	Paper Tape Punch	9	VDU Type 2 *
2	Teleprinter	A	Card Reader
3	Lineprinter	B	Reserved
4	Magnetic Tape	C	Reserved
5	Disc Files	D	Reserved
6	Synchronous Comms.	E	Reserved
7	Asynchronous Comms.	F	General Comms.

QR is the command byte. See DEVICE CONTROL COMMANDS AND PARAMETERS.

* Reserved for use by GEC systems companies.

SEEK

This routine directs the I/O system to locate a nominated record or block. It is only available with random disc files opened for input or update in Random Mode, or with indexed disc files open for input or update in Indexed Mode. The next GET or REPLACE on this stream uses the record or block located by the SEEK.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM
RX	SEQUENCE NUMBER or KEY LENGTH	As on Request
RY	KEY POINTER	As on Request

SEQUENCE NUMBER

Is used with random files and is the sequence number (first is 1) of the record or block that is next to be input.

KEY LENGTH

Is used with indexed files and is the length of the key to be sought.

KEY POINTER

Is not used for random files. For indexed files it is the address of the first byte of the key to be sought.

REPLACE

This routine replaces the current record or block within a disc file. It is available for use with indexed, random and sequential files but only if the stream has been opened in Update Mode. A REPLACE request must be preceded by a GET or a successful SEEK. The user's record or block must be of length equal to the record or block that is to be replaced, unless the file is indexed and accessed in Logical Mode. When a GET precedes a REPLACE the block or record replaced is the same as that read by the GET. When a SEEK precedes a REPLACE the block or record replaced is that located by the SEEK.

Register	Request 0 3 4 7 8 11 12 15	Return 0 1 7 8 15
RA(AM)	0 QUALIFIER	E ERROR CODE
RA(AL)	ID STREAM	ID STREAM
RX	RECORD LENGTH	RECORD LENGTH
RY	BUFFER ADDRESS	BUFFER ADDRESS

RECORD LENGTH

Is the length in bytes of the new record or block.

BUFFER ADDRESS

Is the address of the first byte of the new record or block, with the same restrictions as for GET.

KILL

This routine is only available with DM2. It removes commands from the queue or prevents commands being actioned. The stream is not closed by the KILL command.

If the qualifier is 0 all requests outstanding on the stream are removed. If the qualifier is 1 then only those commands with an ID equal to the RX least significant byte will be removed. Any associated Locate buffers must be freed using Locate 0 when necessary.

Register	Request 0 3 4 7 8 11 12 15			Return 0 1 7 8 15	
RA(AM)	0	QUALIFIER		E	ERROR CODE
RA(AL)	ID		STREAM	ID	STREAM
RX	0		KILLID	0	KILLID
RY	-			-	

KILLID

The ID to be killed if Qualifier = 1. Otherwise ignored.

STATUS

This routine is only available with DM2, and then only with those drivers supporting its use. It provides general information on the driver state, and/or the progress of the current command(s). The qualifier is used to define which information is required. The reply is placed in the user buffer provided.

Register	Request 0 3 4 7 8 11 12 15			Return 0 1 7 8 15	
RA(AM)	0	QUALIFIER		E	ERROR CODE
RA(AL)	ID		STREAM	ID	STREAM
RX	AVAILABLE BUFFER LENGTH			LENGTH OF DEFINED DATA	
RY	BUFFER ADDRESS			BUFFER ADDRESS	

AVAILABLE BUFFER LENGTH

As for GET.

LENGTH OF DEFINED DATA

The number of bytes containing defined information supplied by the driver.

BUFFER ADDRESS

The address of the start of the buffer.

LOCATE

This routine is available only with DM2. It is used to manage locate mode buffers with those drivers supporting this mode of operation. The QUALIFIER defines the exact operation required.

Register	Request 0 3 4 7 8 11 12 15			Return 0 1 7 8 15	
RA(AM)	0	QUALIFIER		E	ERROR CODE
RA(AL)	ID	STREAM		ID	STREAM
RX	LENGTH			LENGTH	
RY	NUMBER/ADDRESS			NUMBER/ADDRESS	

QUALIFIER = 0

1) Obtain a Locate buffer for use with PUT, REPLACE, or EXCHANGE.

   Request LENGTH = undefined

   Request NUMBER/ADDRESS = 0

   Return LENGTH = length of buffer supplied

   Return NUMBER/ADDRESS. Address of buffer supplied. The user must ensure that it is in the appropriate current segment.

2) Release a buffer supplied with GET or STATUS

   Request LENGTH = undefined

   Request NUMBER/ADDRESS = Address of the buffer

   Return LENGTH = 0

   Return NUMBER/ADDRESS = 0

QUALIFIER = 1

   The command should immediately follow a LOCATE mode OPEN, and is used to allocate the Locate segment.

   Request LENGTH = length of each buffer

   Request NUMBER/ADDRESS = Number of buffers required

   Return NUMBER/ADDRESS = Address of the first buffer. This buffer has been allocated for use by the user.

QUALIFIER = 2

   As for QUALIFIER = 1, but no buffers are initially allocated to the user.

EXCHANGE

This routine is only available with DM2, and only with those drivers supporting its use. Its purpose is to provide a convenient method of passing buffers between user and driver. The precise use is driver dependent. Alternate bytes of the buffer define parameter numbers and their values. While setting parameters EXCHANGE often returns the same data.

Register	Request 0 3 4 7 8 11 12 15				Return 0 1 7 8 15	
RA(AM)	0	QUALIFIER			E	ERROR CODE
RA(AL)	ID		STREAM		ID	STREAM
RX	LENGTH FOR OUTPUT				LENGTH OF DEFINED DATA	
RY	BUFFER ADDRESS				BUFFER ADDRESS	

QUALIFIER

Generally used to define the actual operation to be performed on the buffer.

LENGTH FOR OUTPUT

The number of bytes being sent to the driver.

LENGTH OF DEFINED DATA

The number of bytes returned by the driver.

BUFFER ADDRESS

The address of the start of the buffer.

For an overlapped activity on a stream it is necessary to call at least two routines, one to initiate the activity, and another to determine when the activity is complete. All of these routines may also be used with Data Management Type 2.

1) Initiate activity

These routines are analogous to the interlocked routines.

Overlapped Routines	Corresponding Interlocked Routines
OPENO	OPEN
CLOSEO	CLOSE
GETO	GET
PUTO	PUT
CONTROLO	CONTROL
SEEKO	SEEK
REPLACEO	REPLACE
KILLO	KILL
STATUSO	STATUS
LOCATEO	LOCATE
EXCHANGEO	EXCHANGE

The call parameters are identical to those required for the corresponding interlocked routine. If no error is indicated on return then the requested activity has been initiated and its completion must be awaited with a call of type 2). If the error @8909 (indicating Work in Progress) is returned then an activity is already in progress on that stream and its completion must be awaited with a call of type 2). Other errors which may be returned cause the activity not to be initiated so that no call of type 2) is needed.

2) End of activity

These routines are specific to overlapped working and are used to await conditionally or otherwise the result of an activity, or to decode the result of an activity. They are described below. If E is set on return then this indicates an error as specified by ERROR CODE just as if this were the return from a corresponding interlocked routine.

WAITO

This routine is used to cause the user program to be suspended until the outstanding Data Management activity on a particular stream has been completed. It then returns control to the user program yielding the reply for the corresponding activity.

Register	Request		Return	
RA (AM)	0		E	ERROR CODE
RA (AL)	0	STREAM		STREAM
RX	-		reply depends on initial request *	
RY	-		reply depends on initial request *	

The values marked * vary with the requested service. For example, if a 'GETO' request is made, RX contains the record length input and RY contains the buffer address sent: i.e. the reply is the value returned to the standard interlocked version of the preceding request on that stream.

CWAITO

This routine is used to inform the user program whether an outstanding activity on a particular stream has been completed or not. If it has been completed then the reply to the outstanding activity is returned, otherwise the error code @801C is returned.

Register	Request		Return	
RA (AM)	0		E	ERROR CODE
RA (AL)	0	STREAM		STREAM
RX	-		reply depends on initial request *	
RY	-		reply depends on initial request *	

*See command WAITO above.

DECODEIO

When a user process has more than one Data Management activity outstanding, or needs to receive messages from other processes whilst an activity is outstanding and does not wish to be held up with a WAITO or to have to repeatedly call CWAITOs, then the user process should enter the FREE state. When an activity completes on a particular stream the user receives a stimulus in the form of a message on a route corresponding to the stream. The route corresponding to Stream 1 is called IOROUTE and the routes for Stream 2 and so on form a contiguous block from IOROUTE +1 onwards. A message received on one of these routes is of the following form:

Register	Message Parameters	
RA (AM)	* *	
RA (AL)	* *	STREAM
RX	* *	
RY	* *	
RZ	IOROUTE+STREAM-1	

The fields marked * * are reserved.

When it is determined that a message has been received on one of these routes then the routine DECODEIO should be called with the registers set exactly as received with the message except for RZ which is used during the inter-chapter call. DECODEIO decodes the message and returns the registers set exactly as if WAITO had been called for that stream instead of going FREE.

Register	Request		Return	
RA (AM)	* *		E	ERROR CODE
RA (AL)	* *	STREAM		STREAM
RX	* *		reply depends on initial request *	
RY	-		reply depends on initial request *	

*See Command WAITO above.

Notes:

1. When calls are made to WAITO and DECODEIO the segment used as the buffer segment in the initial request must be in the same CST as when the request was made.

2. The user is expected to keep account of which streams are open and which are active.

   @801C Requested service not yet complete i.e. activity not yet completed. Returned by CWAITO.

   @891E No request outstanding, i.e. no activities currently outstanding.

3. It is not permitted to initiate an activity on a DM1 stream on which an activity has not yet been formally completed by means of WAITO, CWAITO or a DECODEIO. However, activities may be initiated on any number of different streams in order to run at the same time.

CONDITIONAL INTERLOCKED/OVERLAPPED ROUTINES

The routines GETCIO, PUTCIO and REPLACECIO have the same interface as the corresponding interlocked routines. They return immediately either with the result of the activity (as for GET, etc.) or with Data Management warning code @801C (c.f. CWAITO), in which case the activity is not yet complete and must be treated as if a GETO, etc, had been called.

This means of working can most profitably replace strict overlapped methods in the case of logical mode access to disc files.

MIXED OVERLAPPED AND INTERLOCKED WORKING ON A STREAM

With overlapped working, each Data Management request OPENO, CLOSEO, GETO, SEEKO, REPLACEO, PUTO, CONTROLO, KILLO, STATUSO, LOCATEO and EXCHANGEO has a corresponding call of CWAITO, WAITO or DECODEIO to obtain the result, unless an error is reported after the initial request. However, sometimes it may be required to operate a stream in overlapped mode but to occasionally use interlocked mode (e.g. use COMGEN to output a message on a stream normally using overlapped mode), or vice versa. This is permitted providing the queue on that stream is not full. Data Management Type 1 supports only a queue length of one and therefore only one activity per stream is permitted. The program simply calls the interlocked Data Management routine instead of the equivalent overlapped routine. The style of the OPEN request (i.e. OPEN or OPENO) is of no consequence to subsequent use.

Care is required when queuing commands to Data Management Type 2 drivers. If using various segments for buffers so queued, the user is responsible for ensuring that the correct segment is loaded while processing data. Also the CST number in RY bits 0-1 always reflects the last command sent. Thus if the user sends buffers from various segments, the contents of RY bits 0-1 will sometimes be incorrect.

RESTRICTIONS WITH CERTAIN MULTIPLEXED DEVICES

With some devices which can have multiplexed input, e.g. teleprinters, it is only possible to direct input to particular processes, not to individual streams of one process. If more than one GET is outstanding on such a device, from the same process, the stream on which input occurs is selected at random from amongst those with outstanding GETs from that process.

Overlapped output to one of these multiplexed devices on more than one stream from the same process results in the outputs being done in random order with no identification as to the correspondence of records with streams.

DM_CONNECT

The routine DM_CONNECT performs stream connection and disconnection and should be called with the following parameters:

Register	Request	
RAM(AM)	0	
RA(AL)	0	STREAM
RX	LENGTH	
RY	Filelist Pointer	

STREAM is the number of the stream to be connected.

LENGTH is the length of the filelist pointed to by the Y-register.

If LENGTH = 0, the stream is to be disconnected.
If LENGTH = 0 and STREAM = 0, all streams are to be disconnected.

The Filelist Pointer specifies a sequence of files to which the stream is to be connected. It should be in a segment with SEND access, and has the following form:

```
                R
<file>[+<file>
```

where <file> is:

```
[a virtual peripheral (spooled or unspooled)
 a temporary disc file
 a context pointer
 a magnetic tape file
 a catalogue disc file ]  [<attribute string>]
```

On return from the routine the registers are set as follows:

Register	Return	
RAM(AM)	STATUS	
RAL(AL)	Filing System ID	STREAM
RX	ACCESS GRANTED	
RY	Filelist Pointer	

Note that the condition register is set to RA.

The value of STATUS varies and has the following meanings:

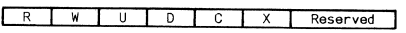
STATUS >=0 means the request was successful. Status > 0 indicates the type of file to which the connection has been made. (Note: The filetype returned is not the same format as that returned by DMCONNECT.)

STATUS <0 means an error has been detected (see below). The ACCESS GRANTED is returned as LENGTH on request.

Filing System ID is as follows:

0 - not a disc file
1 - CFS1 file

ACCESS GRANTED is one halfword, reflecting the access granted to this user:



The letters stand for read, write, update, delete, change attributes and execute access (the last two are for planning purposes only).

Each two bit field specifies the type of access for that category:

Bit Values	Meaning
0	Free access granted
1	Password access granted
2	No access granted
3	Password access not granted

Errors returned as the value of STATUS are either Data Management errors, Catalogue Filing errors with bit 0 set, (@80NN, where NN is the CFS error number) or CONX errors. The meaning of the error codes and the format are described in filing routines or CONX errors, under CATALOGUE FILING ERROR CODES in Facts Book Volume 1, 85-62043.

DMCONNECT

The routine DMCONNECT performs stream connection and disconnection. Attributes such as /Bn, /SCR, /CRE, 5 or 6 character access attributes, /LON or /LOFF, /C or /D are not recognised by this routine but are recognised by DM_CONNECT (see above). DMCONNECT should be called with the following parameters:

Register	Request	
RA(AM)	0	
RA(AL)	0	STREAM
RX	LENGTH	
RY	Filelist Pointer	

STREAM is the number of the stream to be connected.

LENGTH is the length of the filelist pointed to by the Y-register.

The Filelist Pointer specifies a sequence of files to which the stream is to be connected. It must be in a segment with READ and SEND access, and has the following form:

```
                R
<file>[+<file>]
```

where <file> is:

```
[a virtual peripheral (spooled or unspooled)
 a temporary disc file
 a context pointer                                [<attribute string>]
 a magnetic tape file
 a catalogued disc file                          ]
```

On return from the routine the registers are set as follows:

Register	Return	
RA(AM)	E	ERROR
RA(AL)	Undefined	STREAM
RX	LENGTH	
RY	Filelist Pointer	

Note that the condition register is set to RA.

E = 1 If errors have been detected and ERROR = error number

E = 0 If no errors have been detected, ERROR > 0 indicates the type of file to which the connection has been made.

Catalogue filing system errors may also be returned in ERROR, in which case E = 1 and the catalogue filing system error code is returned in bits 1-15 of RA(AL). The meaning of the error codes and the format are described in filing routines or CONX errors, under CATALOGUE FILING ERROR CODES in Facts Book Volume 1, 85-62043.

SPALLOC

Segment space allocation and de-allocation is invoked by calls on the routine SPALLOC. This should be called with the following parameters:

Register	Request
RA(AM)	Sign extension of RA(AL)
RA(AL)	LENGTH
RX	PASTNO

LENGTH determines the amount of space to be allocated as follows:

0< LENGTH < 16384
Allocate LENGTH number of bytes (rounded up to a multiple of 2**SCALEFACTOR bytes).

-16384 = < LENGTH <0
Allocate as much space as possible up to 'minus' LENGTH number of bytes (rounded up to a multiple of 2**SCALEFACTOR bytes). SCALEFACTOR is a system dependent variable.

LENGTH=0 De-allocate space.

PASTNO is the number of the PAST entry of the segment for which space is to be allocated or de-allocated.

Note: The user should ensure that a segment to be de-allocated is not one of the Current Segments.

On return, the registers are set as follows:

Register	Return	
RA(AM)	0	
RA(AL)	0	RESULT
RX	SPACE	
RY	PASTNO	

RESULT = 0 Allocated/De-allocated successfully
1 Segment already allocated to another process
2 Segment already allocated to this process
3 Illegal request
4 Segment already allocated is smaller than request
5 Insufficient space left in partition
6 Insufficient space left in the module allocation
7 Segment pool exhausted
8 Invalid process number
9 Invalid module
10 Invalid partition

SPACE = Length of segment in bytes, for RESULT=0,1,2,4
= Remaining space available in the partition in bytes if RESULT=5
= Remaining space available in the module allowance in bytes if RESULT=6
= PASTNO in all other cases

The Conditions register reflects the value of RESULT.

In OS4000, timer services are provided by the short period Timer module (TIM) which deals in times measure in milliseconds, and the Time of Day module (TOD) which deals in times measured in seconds.

Each module provides an interrogation service (WHEN) which furnishes the current system time in milliseconds from TIM or the current date and time for TOD. In addition TOD may be used to convert a date and time into an age in seconds from time zero on 1 January 1972 and to place this date and time in character form in a user buffer.

Both modules provide timing facilities (EVNT) which can be used to stimulate a user process after a given interval, or repetitively.

Access to the timing facilities is permitted provided the user has entries in his Working Profile corresponding to the required services.

Communication with the appropriate service modules is by way of IPM route names as follows:

Route	Service
WHEN	TOD: WHEN
TODEVENT	TOD: EVNT
TIME	TIM: WHEN
TIMEVENT	TIM: EVNT
DTC	TOD: DTC

GETTIME

The current date and time in seconds may be obtained by calling the routine GETTIME as follows:

Register	Request
RX	LENGTH
RY	BUFFER

If LENGTH is >0, the date and time in character form are put in the user's buffer specified by the Y-register. The format of the complete string that may be placed in the user's buffer is as follows:-

53 26 25 22 21 10 9 1

<System title>	WED	21 JAN 1981	12:34:56

The system title is defined at system generation time as up to 28 characters.

The request LENGTH, if positive, may be from 1 to 53 and indicates how many characters of the complete string, as counted from the right, should actually be placed in the user's buffer.

Values of LENGTH>53 cause the complete string to be placed in the user's buffer starting from address BUFFER just as if LENGTH=53 had been specified. Note that the buffer must be in a segment with write and send access.

If LENGTH<=0 the string is not copied at all.

On return from the routine the date and time are encoded in registers RY and RA:

Register	Return 0 1 6 7 10 11 15			
RA(AM) RA(AL)	TIME			
RX	LENGTH			
RY	0	YEAR AFTER 1972	MONTH IN YEAR	DAY IN MONTH

where:

TIME represents the time of day in seconds.

LENGTH is the number of characters placed in the user's buffer.

FILE MANIPULATION

In the routine descriptions that follow, the registers are set on return according to the following convention:

Register	Return
RA	STATUS
RX	Restored as Request
RY	Restored as Request

where:

STATUS >=0 successful call

< 0 error code in AL bits 1 to 15

The Condition Register is set according to the STATUS returned in RA. The meaning of the error codes and the format are described in filing routines or CONX errors, under CATALOGUE FILING ERRORS in Facts Book Volume 1, 85-62043.

The following terminology is also used:

'FILETITLE'

This is a pointer to:

LONG	TITLE

LONG is a byte giving the number of bytes in TITLE.

TITLE has the form:

$$\left[\begin{array}{l}\text{<name>} \\ \left[\text{<name>}\right]\left[.\text{<name>}\right]^{\circlearrowright}_{1}\end{array}\right]_{1} \quad [\text{<attribute strings>}]$$

The optional attribute string is ignored except by routines CREATEFILE2, CREATE_FILE2, PROTECT_FILE, and CHANGE_ATTRIBUTES.

ARRAY is a pointer to an array which must always start on a full word boundary.

NAME8 is a pointer to a name, filled out to eight characters with zeros.

CREATE_FILE (,ARRAY,FILETITLE)

Creates a file of the specified name with attributes specified in the 48 byte buffer ARRAY.

The ARRAY is formatted as returned by ENCODE_ATTRIBUTES (see ENCODE_ATTRIBUTES).

CREATE_FILE2 (,,FILETITLE)

Creates a file of the specified name with attributes which are the result of merging those attributes specified in the FILETITLE with the default attributes.

DELETE_FILE (,,FILETITLE)

Deletes the specified file.

CREATE_REFERENCE (,FILETITLE1,FILETITLE2)

Creates a reference to FILETITLE2 as FILETITLE1.

DELETE_REFERENCE (,,FILETITLE)

Deletes a reference.

EMPTY_FILE (,,FILETITLE)

Empties the specified file.

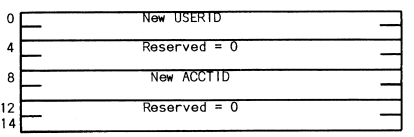
RENAME_FILE (,NAME8,FILETITLE)

Renames the specified file to have the final component NAME8.

SET_FILE_OWNER (,ARRAY,FILETITLE)

Changes the userid and/or the acctid which defines ownership of the file. Requires change userid, quote override id (if changing the userid to SYS) or change acctid privileges.

ARRAY is in the form:

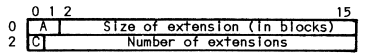


If either the userid or the acctid is not to be changed then the relevant space should be filled with zeros.

EXTEND_FILE (EXTENTS,,FILETITLE)

Extends the specified file.

EXTENTS is an extension capability specification comprising of two halfwords which are considered separately:



A defines the allocation boundary thus:

Value 0 No particular alignment
Value 1-3 Reserved

C defines whether the extensions are to be contiguous. If bit 0 is set, contiguous extensions are required. It is always set if A is not zero.

TRACE_FILE (,,FILETITLE)

Searches to find the file given in FILETITLE and, if found, returns the filetype in RA.

PROTECT_FILE (,,FILETITLE)

The access permission and/or password specified with the FILETITLE is set for the named file.

CREATE_CONTEXT (,NAME8,FILETITLE)

Creates context pointer, NAME8, to specified file.

DELETE_CONTEXT (,,NAME8)

Deletes context pointer specified.

CREATE_PASSWORD (,,NAME8)

Adds specified password to user's list.

DELETE_PASSWORD (,,NAME8)

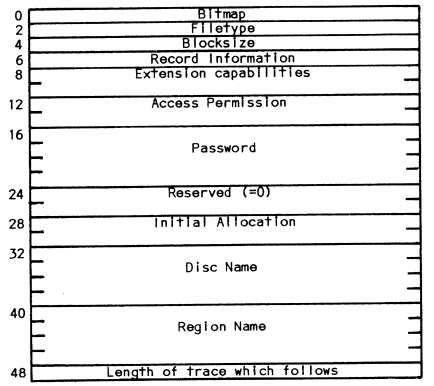
Removes specified password from user's list.

ENCODE_ATTRIBUTES (LENGTH,ARRAY,BUFFER)

Converts an attribute string as used in Job Control Language into the tabular form used by the filing system.

LENGTH is the number of characters in the string.
ARRAY points to a 48 byte area for the encoded attributes.
BUFFER points to the attribute string.

The resulting table is formatted as shown below:



CHANGE_ATTRIBUTES (BITMAP,,FILETITLE)

Allows the simultaneous changing of any genuine attributes specified by the attribute string given with the filetitle and allowed by the value of the bitmap (see the manual System Services, 85-62006).

DECODE_ATTRIBUTES (LENGTH,ARRAY,BUFFER)

This is the inverse of ENCODE_ATTRIBUTES given above. It takes the encoded attributes pointed to by ARRAY and then places the text string describing the defined attributes into the area pointed to by BUFFER (of length LENGTH), returning the length of the string in RX and BUFFER in RY.

SET_GLOBAL_DEFAULTS (LENGTH,,BUFFER)

Sets the default attributes which are to be used globally in the module. The parameters given are as defined in ENCODE_ATTRIBUTES. If the attribute string passed is empty, this is interpreted as a request to reset the appropriate defaults to their state when INIT_FILE_LIB was called.

SET_DEFAULTS (LENGTH,,BUFFER)

This is the same as SET_GLOBAL_DEFAULTS except that the default attributes are used locally for just the current process. If LENGTH is zero then all local defaults are removed (the GLOBAL defaults are all operative).

QUERY_GLOBAL_DEFAULTS (,,ARRAY)

The area pointed to by ARRAY is returned filled with the values of the global default attributes. The format is as for ENCODE_ATTRIBUTES.

QUERY_DEFAULTS (,,ARRAY)

This is the same as QUERY_GLOBAL_DEFAULTS but for local attributes.

INIT_FILE_LIB

This routine records the global defaults. It is advisable to call it before changing the defaults if they are subsequently to be reset to their original state (see SET_GLOBAL_DEFAULTS). This routine also ensures that processes containing FILE_LIB are re-runnable.

INTERROGATE_FILE (LENGTH,ARRAY,FILETITLE)

Interrogate attributes of specified file. These are placed, in encoded form, in the data area pointed to by ARRAY. LENGTH is the number of bytes available for the reply (see System Services, 85-62006).

DEFAULT_USERID (ID)

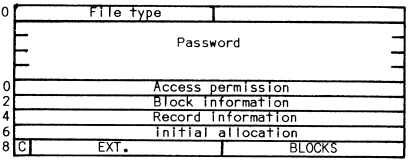
Called to change the user's operative USERID (a privileged operation).

DEFAULT_ACCTID (ID)

Used to change the user's operative ACCTID (a privileged operation).

CREATEFILE (,ATTRIBUTES,FILETITLE)

Creates the specified file with the given attributes. ATTRIBUTES is a pointer to a 20-byte table of the form:



Such a table can be created from a Command Language attributes string by use of the routine ENCODE_ATTRIBUTES. See also routine CREATE_FILE2 below.

The encoding of the fields is as defined in the manual System Services, 85-62006.

DELETEFILE (,,FILETITLE)

Deletes the specified file.

CREATEREFERENCE (,FILETITLE1, FILETITLE2)

Creates a reference to FILETITLE2 as FILETITLE1.

DELETEREFERENCE (,,FILETITLE)

Deletes a reference.

EMPTYFILE (,,FILETITLE)

Empties the specified file.

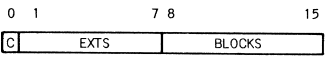
RENAMEFILE (,NAME8,FILETITLE)

Renames the specified file to have the final component NAME8.

EXTENDFILE (,EXTENTS,FILETITLE)

Extends the specified file.

EXTENTS is a 16-bit quantity:



where:

C is 1 if contiguous allocation is required
EXTS is the number of extensions
BLOCKS is the number of blocks in each extension

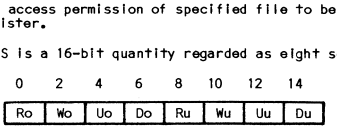
SETFILEPASSWORD (,NAME8,FILETITLE)

Sets password of specified file to the specified name.

SETFILEACCESS (,ACCESS,FILETITLE)

Sets access permission of specified file to be value specified in the X-register.

ACCESS is a 16-bit quantity regarded as eight sets of 2-bit fields:



Suffix 'o' refers to the owner of the file, 'u' refers to other users. R,W,U,D refers to Read, Write, Update and Delete access. A field 'value' of 00 gives Free access, 01 Password access, 10 No access.

CREATECONTEXT (,NAME8,FILETITLE)

Creates context pointer, NAME8, to specified file.

DELETECONTEXT (,,NAME8)

Deletes context pointer specified.

CREATEPASSWORD (,,NAME8)

Adds specified password to user's list. The array containing the name must be FULL word aligned and further the segment containing the array must have Write access as the password scrambling is done in situ.

DELETEPASSWORD (,,NAME8)

Removes specified password from user's list.

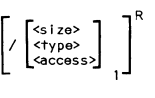
ENCODEATTRIBUTES (LENGTH, ARRAY, PTR)

Converts an attribute string as used in Job Control Language into the tabular form used for CREATEFILE, SETFILEACCESS etc.

LENGTH is the number of characters in the string.
ARRAY points to a 20 byte area for the encoded attributes.
PTR points to the string

The routine combines the specified attributes with the default attributes. These initially correspond to /LST/FFFF.FNNN/NORMAL, but may be changed by the routine SETDEFAULTATTRIBUTES.

The attributes string has the format:



and is terminated by reaching the end of the length specified or by reaching a zero byte.

<type> is a file format attribute.

<size> is a file size attribute.

<access> is a file protection attribute.

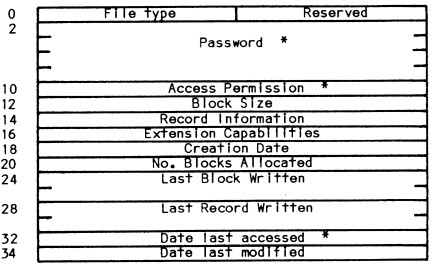
SETDEFAULTATTRIBUTES (LENGTH,,PTR)

Parameters as for ENCODEATTRIBUTES. Changes the defaults to agree with the specified attributes.

The attributes set as a result of using this routine affect the current run-time environment of the user program only and in no way relate to the attributes known to the user Command Process.

INTERROGATEFILE (,ARRAY,FILETITLE)

Discovers attributes of specified file. These are placed, in encoded form, in the 36-byte data area pointed to by ARRAY. Password and access permission are returned as 0 if the requestor is not the owner of the file. The format of the attributes is:



The fields marked * (and RA) are returned holding the value 0 if the user's currently set USERID is not that with which the file was created.

CREATEFILE2(,,FILETITLE)

Creates a file with attributes which are the result of merging the attributes specified in the FILETITLE with the default attributes.

DEFAULTUSERID (ID)

Sets the default USERID to ID. This is a privileged operation.

DEFAULTACCTID (ID)

Sets the default ACCTID to ID. This is a privileged operation.

ROUTINES FOR DEALING WITH SEND MESSAGES

ENABLESEND

This routine must be called if a SEND message is to be accepted from the Command Process. When a message has been received, it should be called again if a further message is to be permitted.

Register	Request	Return
RX	LENGTH	Undefined
RY	BUFFER	Undefined

LENGTH Maximum length of message acceptable (RX=0 is permissible). The message sent is truncated to this length.

BUFFER Pointer to buffer for the message if LENGTH>0. The buffer should be in CST0, CST1 or CST2.

ENABLEWARN

This routine must be called if a message is to be received from the command process in the event of a warning being sent to it - either because of time exhaustion or operator action. If the command process has already received a warning when the routine is called a message is sent immediately.

Register	Request	Return
RA	Undefined	As on Request
RX	Undefined	As on Request
RY	Undefined	As on Request

AWAITSEND

This routine awaits a message generated by the SEND command or a warning received by the command process. Note that ENABLESEND must previously have been called if a message is to be received and ENABLEWARN must previously have been called if a warning is to be received.

Register	Request	Return
RX	Undefined	LENGTH
RY	Undefined	POINTER

LENGTH
- -2 Warning (overrides any other message)
- -1 No message can be received (neither ENABLEWARN nor ENABLESEND has been called)
- >=0 Length of message

POINTER Pointer to message

CHECKFORSEND

This routine checks for a message generated by a SEND command or a warning received by the command process, but does not wait for one to arrive if none is present. Note that ENABLESEND or ENABLEWARN must previously has been called for the presence of a message to be tested.

Return parameters are as for AWAITSEND, except that LENGTH=-1 if no message is present. The Conditions Register is set according to the value in RX.

GETPROCNO

This routine translates a printable form of a process name into its process number. The name is in the form:

```
<module name>.<process name>
or  <process name>
```

in which case the user's own module is assumed.

Register	Request	Return
RA	Undefined	VALUE
RX	LENGTH	LENGTH
RY	BUFFER	BUFFER

LENGTH Length of name

BUFFER Pointer to buffer containing the name

VALUE Process number, if process found, otherwise:

-1 if process unknown
-2 if BUFFER is invalid
-3 if user has insufficient privilege to obtain the process number.

GETPROCNAME

This routine converts a process number into a printable form of a process name in a buffer supplied by the user.

Register	Request	Return
RA	PROCNO	VALUE
RY	BUFFER	BUFFER

PROCNO Process number of required process

BUFFER Pointer to a 9-byte buffer in a writable segment

VALUE Length of name, if no errors
-1 if process number is invalid
-2 if BUFFER is invalid
-3 if user has insufficient privilege to obtain the process name

GETVAR

This routine obtains the value of an integer-valued system or user-defined JCL variable or constant.

Register	Request	Return
RA	Undefined	STATUS
RX	LENGTH	VALUE
RY	BUFFER	BUFFER

LENGTH Number of characters in variable name

BUFFER Pointer to variable name

STATUS
- 0 if variable found
- -1 if variable unknown or wrong type
- -2 if BUFFER is invalid

VALUE Value of variable

The Conditions Register is set according to the STATUS returned in RA.

SETVAR

This routine gives a system or user variable, which may or may not have been defined previously, an integer value.

Register	Request	Return
RA	VALUE	STATUS
RX	LENGTH	Undefined
RY	BUFFER	BUFFER

VALUE Value to be given to the variable

LENGTH Number of characters in variable name

BUFFER Pointer to variable name

STATUS
- 0 if variable set
- -1 attempt to set system constant or change type of system variable
- -2 if BUFFER is invalid
- -3 if the variable dictionary is full

The Conditions Register is set according to the STATUS returned in RA.

GETSTRINGVAR

This routine obtains the value of a string-valued JCL variable.

Register	Request	Return
RA(AM)	REPLYLENGTH	STATUS
RA(AL)	REPLYPOINTER	
RX	LENGTH	STRINGLENGTH
RY	BUFFER	REPLYPOINTER

REPLYLENGTH Length of buffer REPLYPOINTER. For convenience, if REPLYLENGTH is merely a sign-extension of REPLYPOINTER, a length of 80 is assumed.

REPLYPOINTER Pointer to a buffer to receive the value of the variable.

LENGTH Length of variable name

BUFFER Pointer to variable name

STATUS
 0 if variable found
-1 if variable unknown or wrong type
-2 if either buffer invalid
-4 if REPLYLENGTH too short

STRINGLENGTH Length of string value of the variable

The Conditions Register is set according to the STATUS returned in RA.

SETSTRINGVAR

This routine gives a user variable, which may or may not have been defined previously, a string value.

Register	Request	Reply
RA(AM)	STRINGLENGTH	STATUS
RA(AL)	STRINGBUFFER	
RX	LENGTH	LENGTH
RY	BUFFER	BUFFER

STRINGLENGTH Length of string to be assigned to the variable. Must not exceed 252.

STRINGBUFFER Pointer to a buffer containing the string to be assigned to the variable. Must be in same segment as BUFFER.

LENGTH Length of variable name.

BUFFER Pointer to variable name. Must be in same segment as STRINGBUFFER.

STATUS
 0 if variable set
-1 if string too long or attempt to change type of system variable
-2 if buffers are invalid
-3 if the variable dictionary is full

The Conditions Register is set according to the STATUS returned in RA.

GETARG

This routine obtains the value of any argument of the command which invoked the process.

Register	Request	Return
RA(AM)	REPLYLENGTH	STATUS
RA(AL)	REPLYPOINTER	
RX	LENGTH	VALUE
RY	BUFFER	REPLYPOINTER

Note: For repeated arguments, successive calls of the routine for a specific argument name yields successive values until the set is exhausted, then STATUS=-1 is returned. For a non-repeated argument, STATUS=-1 is returned on the second and subsequent calls.

REPLYLENGTH Length of REPLYPOINTER. For convenience, if REPLYLENGTH is simply a sign-extension of REPLYPOINTER, a length of 80 bytes is assumed.

REPLYPOINTER Pointer to a buffer for the argument string if it is of type STREAM, FILE, STRING or NONE.

In the case of a STREAM argument, the first byte of the argument string is the stream number, and this is followed by the corresponding filelist with default attributes merged. For a FILE argument, the argument string is the file title with default attributes merged. For a STRING argument, the argument string is the string concerned with quotes removed. For a NONE argument, the argument string is the remainder of the Command Line. For a PROFORMA argument, the argument string is the proforma name.

LENGTH Length of argument name

BUFFER Pointer to argument name

STATUS
-1 If no more arguments of specified name are present
-2 If either buffer is invalid
-4 If REPLYLENGTH is too short

>=0 Type of argument:

0 = STREAM
1 = FILE
2 = STRING
3 = BOOL
4 = VALUE
5 = NONE
6 = PROFORMA

VALUE Value of argument if BOOL or VALUE type, otherwise length of argument string in REPLYBUFFER. Possible values are:

BOOL 0 if FALSE, -1 if TRUE
VALUE -32768<=value<=32767

GETSTREAMARG

This routine obtains the value of the argument used to specify the connection of a stream of the process.

Register	Request	Return
RA	STREAM	STATUS
RX	LENGTH	REPLYLENGTH
RY	BUFFER	BUFFER

STREAM — Number of the stream

LENGTH — Length of the buffer for reply

BUFFER — Pointer to buffer for reply

STATUS
- -4 If buffer too short (first LENGTH bytes are returned)
- -1 No such argument
- 0 Successful

REPLYLENGTH — Length of reply

ROUTINES TO OBTAIN STANDARD ERROR MESSAGES

GETDMERR

This routine obtains a standard error message for a Data Management error.

Register	Request	Return
RA	ERRORCODE	STATUS
RX	Undefined	LENGTH
RY	BUFFER	BUFFER

ERRORCODE — Value return in AM by Data Management but positioned in AL.

BUFFER — Pointer to a buffer of at least 80 bytes in length.

STATUS
- -1 if unknown ERRORCODE
- -2 if invalid BUFFER
- 0 otherwise

LENGTH — Length of message in BUFFER

The Conditions Register is set according to the STATUS returned in RA.

GETCFERR

This routine obtains standard messages for error numbers returned by the Catalogue Filing System or Data Management stream connection.

Register	Request	Return
RA	ERRORCODE	STATUS
RX	Undefined	LENGTH
RY	BUFFER	BUFFER

ERRORCODE Error number

BUFFER Pointer to a buffer of at least 80 bytes in length.

STATUS
-1 if unknown ERRORCODE
-2 if invalid BUFFER
0 otherwise

LENGTH Length of message in BUFFER

The Condition Register is set according to the STATUS returned in RA.

GETERROR

This routine obtains standard error messages when supplied with an error number and class.

Register	Request	Return
RA	ERRORCODE	STATUS
RX	ERRORCLASS	LENGTH
RY	BUFFER	BUFFER

ERRORCODE Error number

ERRORCLASS is the class of the error. Bit 8 of ERRORCLASS, if set, indicates that the service detecting the error was a remote computer. Bit 9 of ERRORCLASS, if set, indicates that a secondary error code is available. This error code is found in bytes 1 to 4 of the buffer. Bits 10 to 15 identify the service as follows:

ERRORCLASS Class of error (in bits 9-15):

0	Catalogue Filing	8	ISPL
1	Data Management	9	RAL
2	EMA*	10	Job Description Errors
3	ACCO	11	BS
4	STOR	12	CMNI
5	PCON	13	ICI
6	BLDR	14	TLOG
7	DSPL	15	Process errors

* If RA(AM)>0 errors are detected by the service for which RA(AM) is the errorclass.

BUFFER Pointer to a buffer at least 88 bytes long. On return byte 0 contains the length of the text returned, not including itself.

STATUS
-2 If invalid BUFFER
-1 If invalid ERRORCODE
0 Otherwise

LENGTH Length of text in BUFFER, not including byte 0.

ROUTINES TO INTERROGATE TIMER VALUES

GETSTEPCOMP

This routine requests the value of STEPCOMP remaining for this process.

Register	Request	Return
RA	Undefined	STEPCOMP

STEPCOMP is the fullword value of computation time left in IEUs.

GETSTEPEXEC

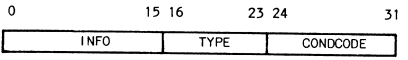
This routine requests the value of STEPEXEC remaining for this process.

Register	Request	Return
RA	Undefined	STEPEXEC

STEPEXEC is the fullword value of execution time remaining in minutes.

CONDITION CODE AND MESSAGES ON STOP INSTRUCTION

When a process executes a STOP instruction, the Command Process interprets the value in the A-register as follows:



TYPE	Action
0	Set CONDCODE System Variable
1	Set CONDCODE System Variable and produce Data Management error message for error number INFO
2	Set CONDCODE System Variable and produce JCL error message for error number INFO
3	Set CONDCODE System Variable and produce Catalogue Filing error message for error number INFO
Others	Set CONDCODE System Variable. Reserved.

The JCL system variable ERRTYPE is set to TYPE; ERRVAL is set to INFO.

Two radix conversion modules are available:

1) RADIXCONVERSION, a program chapter containing:

   - FROMCHAR, a routine which converts numbers in character form to their binary representation.
   - TOCHAR, a routine which converts binary numbers into their character representation.

   The radix conversion routines are capable of handling both integer and real numbers.

2) FCHARS, a code conversion table which is a Data Chapter used by FROMCHAR in converting characters to their internal binary form. This is automatically linked from the library if FROMCHAR is used.

CHARACTER STRING TO BINARY CONVERSION (FROMCHAR)

FROMCHAR provides conversion of strings of characters in a user's buffer to binary values. The string may be one of the following forms:

- Decimal integer
- Decimal real
- Real with exponent
- Hexadecimal
- Binary

One of the calling parameters is used to specify whether one particular form is expected or whether any form is acceptable. In this latter circumstance, hexadecimal and binary character strings must be preceded by the appropriate symbol: @ for hexadecimal, % for binary.

FROMCHAR detects the end of the character string by reference to a length parameter supplied by the user or by encountering an invalid character. A full description of the formats allowed by FROMCHAR is provided under NUMBER FORMATS below.

The call and return interfaces for FROMCHAR are as follows:

1) Call, Integer Conversion

Register	Request	Return
RA		NUMBER
RX	'Options'	Error Indications
RY	Buffer Address	Return Buffer Address

2) Real Number Conversion

Register	Request		Return
RA(AM)	Not Used		Not Used
RA(AL)	'Scale'	'Places'	Floating Point Address
RX	'Options'		Error Indications
RY	Buffer Address		Return Buffer Address

The return buffer address points to the character in the user's buffer following the last valid character processed. Full definitions of 'options', 'scale' and 'places' are provided under FROMCHAR CALL PARAMETERS below.

The 'options' specified in the X-register on calling FROMCHAR consists of three parts:

1) OPT (Bits 0-3)

2) MODE (Bits 4-7)

3) LENGTH (Bits 8-15)

The OPT field can take the values 0, 1 or 2 only; all other values result in an error being returned. OPT is set to 1 to provide the FORTRAN option. OPT equals 2 to provide the Fortran 77 option.

- When the FORTRAN option is operative, non-leading spaces are treated as zero.
- When the FORTRAN 77 option is operative, all spaces encountered in reading numbers are ignored unless the field contains only spaces, in which case the value of the field is treated as zero.
- When both these options are operative:
  - The real exponent symbol E may optionally be replaced by the + or - sign.
  - The least significant half of the A-register (AL) contains two further parameters SCALE (bits 0-7) and PLACES (bits 8-15).

SCALE is treated as an 8-bit signed integer n, which, in the absence of an exponent value in the input number, is used to 'scale' the number, by multiplying the number by 10**-n. Thus, for example if SCALE=1 an input of 10.34**-2 would result in 0.1034 being returned (specified exponent -2 overrides 'SCALE') whilst an input of 10.34 would result in 1.034 being returned (i.e. 10.34x10**-1).

PLACES is treated as an 8-bit unsigned integer which, in the absence of an explicit decimal point in the input number, is used to define the number of digits in the fractional part. Thus, for example, if PLACES=3 an input of 31416 would result in 31.416 being returned, whilst 3.1416 would be returned as such because of the presence of the decimal point.

MODE is used to specify the format and radix of a number as follows:

```
MODE=0  Signed or unsigned decimal integer
    =1  Unsigned hexadecimal integer
    =2  Unsigned binary integer
    =4  Signed or unsigned real number
    =7  Any of the following:
        signed or unsigned decimal integer
        signed or unsigned real number
        unsigned hexadecimal integer immediately preceded by '@'.
        unsigned binary integer immediately preceded by '%'.
```

LENGTH specifies the maximum number of characters in the user's buffer which may be scanned for a number.

NUMBER (RA)

If the call on FROMCHAR produces an integer result, its value is held in the A-register.

Floating Point Address (AL)

If the result is a floating point number, the A-register holds the address of a double precision location in FROMCHAR local workspace containing this value. The FLAGS field may be examined to determine if the number contained an E or D exponent.

ERR (RX Bits 0-3)

This value indicates the occurrence of conversion errors as follows:

ERR = 0 No errors
= 2 Number reported for binary or hexadecimal conversion is too large.
= 3 Invalid number. A termination character was encountered before any significant digits were found.
= 4 No number found. Entry was made with LENGTH=0 or only spaces were found in the buffer.
= 14 Invalid MODE or OPT
= 15 MODE not available

The A-register is set to 0 on any error except ERR=3.

After ERR=2 has been returned, for real numbers the value of the double word pointed to by RA is undefined, otherwise the integer value returned in RA is the value of the number obtained by ignoring the most significant digits of the character representation until the number remaining falls within range.

After ERR=3 has been returned, FLAGS indicates whether a sign has been encountered. The A-register contains the class value of the offending character (see CODE CONVERSION TABLE below) and Return Buffer Address points to its position in the user's buffer. Remaining Length is suitably updated.

In the case of the remaining errors, RA=0 on return from FROMCHAR, except for ERR=4 in MODE 4 when RA points to a real number of value 0.

FLAGS (RX Bits 4-7)

This value is to be regarded as a set of flags indicating the nature of the return value:

bit 4 = 0 Integer value
= 1 Real value
bit 7 = 0 Unsigned number
= 1 Signed number

For integer values:

bits 5-6 = 10 Binary integer found
= 01 Hexadecimal integer found
= 00 Decimal integer found

For real values:

bit 5 = 0 No exponent, or E exponent found (i.e. single precision floating point number).
= 1 D exponent found (i.e. double precision floating point number).

Real values are always returned by FROMCHAR in floating point form.

REMAINING LENGTH (RX Bits 8 - 15). This is LENGTH decremented by the number of characters processed by FROMCHAR.

RETURN BUFFER ADDRESS (RY). This is a pointer to the character in the user's buffer following the last valid character processed.

CONDITIONS REGISTER. The Conditions Register is set by the value returned in the X-register.

TOCHAR provides conversion of binary values to their character representation as decimal, hexadecimal or binary integers or as real numbers in a user's buffer. The characters representing the number are aligned against the right-hand end of the specified field. The characters generated by TOCHAR are signs, spaces, the decimal digits, the decimal point, and the hexadecimal digits A-F, in upper case. The TOCHAR call and return parameters are passed in the registers as follows:

1) Integer Conversion

Register	Request	Return
RA	NUMBER	
RX	Options	Error Indications
RY	Buffer Address	Return Buffer Address

2) Real Number Conversion

Register	Request	Return
RA(AM)	SCALE | PLACES	
RA(AL)	Address of Real Number	
RX	Options	Error Indications
RY	Buffer Address	Return Buffer Address

These parameters are fully described under TOCHAR RETURN PARAMETERS.

TOCHAR is capable of handling either integer or real numbers, but the call parameters passed in the A-register depend upon which type of number is being handled. This section gives details of call parameters shown above. Since the use of the X and Y-registers is the same for both integer and real numbers, these registers are dealt with first.

X-Register

The 'options' specified in the X-register on calling TOCHAR comprise three parts:

1) OPT (Bits 0-3)

2) MODE (Bits 4-7)

3) LENGTH (Bits 8-15)

Dealing with these in reverse order:

The LENGTH field specifies the width of the user's field.

The MODE parameter specifies the form and radix of the conversion, as follows:

MODE = 0	Decimal integer		
= 1	Hexadecimal integer		
= 2	Binary integer		
= 4	Floating point (E exponent)	Single	precision
= 5	Fixed point		argument
= 6	Free point		
= 12	Floating point (D exponent)	Double	precision
= 13	Fixed point		argument
= 14	Free point		

The OPT parameter controls the application of formatting options, as shown below:

OPT Bit number	MODE	Bit Unset	Bit Set
0	All	Leading spaces	Leading zeros
1	All	Floated sign	Fixed sign
2	All	Unsuppressed sign	Suppressed sign
3	All except 1 & 2 (decimal)	Signed:blank plus-sign	Signed:explicit plus-sign
	1 or 2 (hexadecimal or binary)	Unsigned:sign character is @ or %	Signed:explicit plus-sign

TOCHAR RETURN PARAMETERS

On return from TOCHAR a negative value in the X-register indicates that an error has occurred. More precisely, Bits 0-3 are used for an error indication and Bits 8-15 return the length of the buffer used for the conversion.

Note that register A is undefined on return.

ERR (RX Bits 0-3)

This parameter indicates the occurrence of conversion errors as follows:

ERR = 0 No errors

= 8 Insufficient field width for conversion. The specified field is filled with the partially converted number and RETURN ADDRESS points to the next available position after the field.

= 9 Invalid form for a floating point number detected.

= 10 Scale factor for floating point mode does not lie in the range -d=<n=<d+1.

= 14 Invalid MODE

= 15 MODE requested not available from the particular copy of TOCHAR in use.

LENGTH (RX Bits 8-15) is returned unchanged.

The Return Buffer Address parameter is the user's calling buffer address incremented by LENGTH. It thus points to the next unfilled element of the user's buffer.

The Conditions Register is set by the value returned in the X-register.

CODE CONVERSION TABLE (FCHARS)

In FROMCHAR, the 7-bit input code characters are separated into classes, and in the event of an error the class value is returned in the A-register (see below).

The values are held in an 128-byte array indexed by character code value which forms the Data Chapter FCHARS.

CHARACTER CODE CLASS FORMATS

Character	Class Value	Comment
0-9	0-9	
A-Z	10-35	Upper and lower case treated as equivalent
a-z	10-35	
<space>	128	
, <comma>	129	
@	130	
+ <plus>	131	
- <minus>	132	
%	133	
. Full stop (decimal point)	134	
! <exclamation mark>	135	
All others	255	Invalid characters

The rules under which numbers are converted by FROMCHAR are as follows:

- Leading spaces are ignored.
- A number is terminated by any inappropriate character or by the end of the user's buffer.
- A decimal integer or real number may have an optional preceding + or - sign.
- An integer is a non-empty string of appropriate digits:

  0 or 1 (binary)
  0 to 9 (decimal)
  0 to 9 A to F, a to f (hexadecimal)

- A basic real number has the form:

  <integer part>.<fractional part>

  where <integer part> and <fractional part> are decimal strings, one of which may be null.

- A real number has one of the forms:

  <basic real number>
  <basic real number> E <optionally signed decimal integer>
  <decimal integer> E <optionally signed decimal integer>

  Additionally, if MODE=4 (see FROMCHAR REQUEST PARAMETERS) a decimal integer will be converted to a real number.

- In real numbers, E may be replaced by D for double precision.
- Under the FORTRAN option (see FROMCHAR REQUEST PARAMETERS) non-leading spaces are treated as zero whereas the Fortran 77 option ignores non-leading blanks. A field which is entirely blank will be regarded as zero.

  The form:

  E <signed decimal integer>

  may be replaced by the signed decimal integer, for example:

  1-10 = 1E-10 or
  1+10 = 1E+10

- There is a maximum permitted number of digits in a real number (normally 63). If this number is exceeded, Error 2 will be returned. The minimum accuracy of GEC 4000 Series computers in holding a real number is:

  Single precision 1 part in 2.10x10**6
  Double precision 1 part in 9.01x10**15

  Under no circumstances is it meaningful to provide as input a decimal string of more than 17 characters.

Note: All combinations of bit pattern are meaningful, though not all are useful.

If there is insufficient space in the buffer to convert a positive number with a floated blank plus sign, the sign character is suppressed.

Option	Meaning
Leading spaces	The buffer is padded out with spaces.
Leading zeros	The buffer is padded out with zeros.
Floated sign	The sign character immediately precedes the number.
Fixed sign	The sign character appears at the extreme left hand side of the buffer.
Unsuppressed sign	The sign character is present.
Suppressed sign	The sign character is absent.
Blank plus sign	If the number is non-negative the sign character is a space.
Explicit plus sign	If the number is non-negative the sign character is a + character.
Signed	In modes 0, 1 and 2 the A-register is treated as a 32-bit signed integer in the range:-2**31 to (2**31)-1. In all other modes the real number is signed.
Unsigned	The A-register is treated as a 32-bit pattern. When present, the sign character is a space.

DEFINITION OF TERMS

Y-Register

The Y-Register contains the address of the buffer into which the characters resulting from the conversion are to be placed.

A-Register

For integer conversion, the A-register (AM and AL) holds the number to be converted.

When converting real numbers, however, the most significant half of the register (AM) is used to hold two 8-bit parameters. These are:

SCALE (Bits 0-7) and PLACES (Bits 8-15). SCALE is taken to be an 8-bit 2's-complement signed integer, n, to be used as a scale factor with the following effect:

Floating point (MODE=4 or 12)	The basic real component of the converted number is multiplied by 10**n and its exponent is then decremented by n. A number converted with scale factor zero will have no digits before the decimal point, and no leading zeros after the point.
Fixed point (MODE=5 or 13)	The number is multiplied by 10**n before conversion.
Free point (MODE= 6 or 14)	The scale factor has no effect unless the number is converted in floating point form.

PLACES is taken as an 8-bit positive integer, d, which is used to determine the number of digits to be placed after the decimal point. The number is rounded to this precision by adding 5 in the next decimal place. Note that the user must ensure that the total field width is sufficient to contain this and the other components of the number (i.e. sign, integer part, decimal point and exponent part).

The least significant half of the A-register (AL) is used to hold the address in the user's workspace containing the value to be converted.

It will be seen that the conversion of real numbers involves three modes:

- Floating Point
- Fixed Point
- Free Point

and that each of these types can be either single precision or double precision.

The resultant format depends also upon three other parameters:

- Scale Factor, n
- Requested number of places, d
- Conversion field width, w

The effect of these parameters for each of the three modes is given below:

- Floating Point Mode

  There are two cases, depending on the sign of n.

  n>0 There are n digits before the decimal point and d-n+1 after the point.

  n<=0 There are n zeros after the decimal point followed by d+n significant digits.

  This is followed by the letter E or D according to the precision of the argument supplied, a sign and two digits of exponent value. The exponent is reduced by n. The scale factor, n, must lie in the range -d<=n<=d+1.

- Fixed Point Mode

  The value to be converted is multiplied by 10**n. It is converted with d digits after the decimal point.

- Free Point Mode

  The format of numbers converted in this manner depends on the magnitude of the number, N as follows.

0.1 <=N <1	d	digits	after the decimal point.
1 <=N <10	d-1	digits	
10**(d-1) <=N <10**d	0	digits	

If N is not in any of the above ranges, the number is converted in floating-point format. For alignment reasons, the fixed point conversions are followed by four spaces. The scale factor is only effective if the floating-point style is adopted.

ERROR REPORTING

In OS4000 errors which are detected by Data Management are classified into two categories:

- Inferior Errors (Bit - 4 of the ERROR CODE is unset)

  These are errors which are not caused by the user process doing anything illegal.

- Superior Errors (Bit - 4 of the ERROR CODE is set)

  These are errors caused by the user process attempting something illegal.

In addition to these two categories there are eight different classes of error (0-7), and the error class is set in bits 5, 6 and 7 of the ERROR CODE.

Class 0 - Normal Status

Class 1 - Parameter Errors

Class 2 - Device Off-Line

Class 3 - Device Not Available

Class 4 - Status Indication

Class 5 - Device Failure

Class 6 - Data Errors (Device specific)

Class 7 - Buffer Irregularities

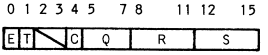
Class 7 is used for errors concerning buffer lengths that may be processed but where some part of the record may be lost.

Due to the large number of possible error indications and the fact that some of them may occur together, errors are grouped into two types:

1) Mutually exclusive (Bit-1 of the ERRORCODE unset).

2) Non-mutually exclusive (Bit-1 of the ERRORCODE is set).

Data Management errors are described by a halfword of information:



E : determines the error code:
Unset - No error (Bits 1 to 15 undefined)
Set - Errors as defined by bits 1 to 15 (the ERROR CODE)
T : determines the error type:
Unset - Type 0 error
Set - Type 1 error
C : determines the error category:
Unset - Inferior error
Set - Superior error
Q : determines the error class.
R, S: contain the error number or bits.

The error tables that are contained in Facts Book Volume 1 are in numerical order of the error numbers ignoring C and Q. However, there are two tables, one for each 'type' of error.

To use the tables proceed as follows:

1) If T = 0 (ERROR CODE starts with 8 or 9) look at the Error Type 0 table to find your error.

   If ERROR CODE has the form 93RS, the error is one returned to Data Management by the Catalogue Filing System. See Catalogue Filing System Errors in Facts Book Volume 1, 85-62043 for the interpretation of RS.

2) If T = 1 (ERROR CODE starts with hex digit C), look at the Error Type 1 table. This contains a number of different types of error but as the error code can contain one or more errors, they are found by the following method:

   a) Consider R and S as a bit pattern.

   b) If the bit pattern is as defined in the table then that is the error.

   c) If the bit pattern is a summation of a number of error codes then each is an error. For example:

      C610 - Block Sequencing Error

      C605 - C601 (Parity or cyclic redundancy check error)
             C704 (Input record longer than user's buffer).

      Note: The error class Q, need only correspond with one of the error codes (6 in the example above).

At OPEN time the user may select whichever action the system is to take for each class of error. In the event of not making such a specification, the system provides defaults suitable for most user processes.

The Error Option is part of the OPEN command. The default is obtained by leaving RX bit-0 as zero. If RX bit-0 is 1 then RY defines a halfword option. Class 0 errors are always returned to the user process and not to its owner.

Meaning	Error Option Bits Set In RY															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Suppress error message to owner for error class indicated	-		1		2		3		4		5		6		7	
Return error to user for error class indicated		-		1		2		3		4		5		6		7

Pairs of bits are used to control action on a given error class. The class number is controlled by a given effect. In this case:

00 Report error to owner but not to user process
01 Report error to owner and return error to user process
10 DO NOT USE
11 Return error to user process only

DEFAULT ERROR HANDLING

When bit-0 of RX is not set the system provides default error handling as follows:

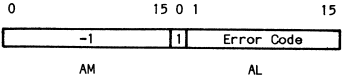
- Logical working

  The default is equivalent to error option @C000 so that class 0 errors are returned only to the user process while all other classes are reported only to the owner.

- Physical working

  The default is equivalent to error option @C0FF so that errors of classes 1-3 are reported only to the owner while errors of classes 0, 4-7 are returned only to the user process.

Errors are reported by the routines with AM= -1, AL bit 0=1 and a code in AL bits 1 to 15:



For description of error codes see Facts Book Volume 1, 85-62043.

Two routines have been introduced to deal with identifiers of up to six alphanumeric characters.

Rules for expansion and compression are:

- Six character identifiers are compressed to fit into a fullword.
- Five character identifiers are extended to six by the appending of a null character before being compressed to fit into a fullword.
- Identifiers of four characters or less need no compression to fit into a fullword.

PACK

Register	Request	Return
RA(AM) RA(AL	Undefined	Compressed Identifier
RX	LENGTH	LENGTH
RY	POINTER	POINTER

LENGTH is the length of the identifier to be compressed. It must not be greater than six.

POINTER is a pointer to the identifier string. It must point to a valid alphanumeric string.

Compressed Identifier is either the unaltered identifier (if of four characters or less), or the five or six character identifier compressed to fit into a fullword.

UNPACK

Register	Request	Return
RA(AM) RA(AL)	Compressed Identifier	Undefined
RX	Undefined	6
RY	POINTER	POINTER

Compressed Identifier is the identifier to be expanded.

POINTER is a pointer to a 6-byte buffer to hold the expanded identifier. It should be in a segment with write access.

6 is the length of the returned identifier string.

The main GEC 4000 Series language processors convert their source code modules into object module form. The function of the Linkage Editor is to combine these object modules to form a process module (i.e. object program) suitable for loading and execution.

LINKING WITHOUT COMMANDS

The facility of linking without specifying any commands is provided primarily for users wishing to run programs without making explicit use of the GEC 4000 Series underlying process structure. System defaults automatically apply upon the absence of specific Linkage Editor commands, which are listed below.

```
DEFAULT
ENTRY ENTRYPOINT
DEBUG
MAP
ROUTES CSALLOC WHEN CONN TOMULTI FILE SUPV TIME TOSMA,
CHAP TOLOAD PCON TODSW DTC TIMEVENT *14 OPER *16 SUBSYSTEM,
ACCO *22 TOTAL IOROUTE
!PAST   1
PAST LOCAL,,
!PAST   3 : 8  code segments
CODE1  CODE2  CODE3  CODE4  CODE5  CODE6  CODE7,
CODE8,
!PAST 11: 20 data segments
DATA1  DATA2  DATA3  DATA4  DATA5  DATA6  DATA7,
DATA8  DATA9  DATA10 DATA11 DATA12 DATA13 DATA14,
DATA15 DATA16 DATA17 DATA18 DATA19 DATA20,
!PAST 31 : 12 streams + CSDUMP
STREAMS *43 CSDUMP
! PAST 44 : 14 EMPTY segments
! PAST 58 :  3 TOTAL segments
*58 TOTAL1 TOTAL2 TOTAL3,
! PAST 61 : 4 library work segments
LIBWORK1 LIBWORK2 LIBWORK3 LIBWORK4,
! PAST 65 : 30 code segments
CODE9  CODE10 CODE11 CODE12 CODE13 CODE14 CODE15,
CODE16 CODE17 CODE18 CODE19 CODE20 CODE21 CODE22,
CODE23 CODE24 CODE25 CODE26 CODE27 CODE28 CODE29,
CODE30 CODE31 CODE32 CODE33 CODE34 CODE35 CODE36,
CODE37 CODE38,
! PAST 95 : 50 data segments
DATA21 DATA22 DATA23 DATA24 DATA25 DATA26 DATA27,
DATA28 DATA29 DATA30 DATA31 DATA32 DATA33 DATA34,
DATA35 DATA36 DATA37 DATA38 DATA39 DATA40 DATA41,
DATA42 DATA43 DATA44 DATA45 DATA46 DATA47 DATA48,
DATA49 DATA50 DATA51 DATA52 DATA53 DATA54 DATA55,
DATA56 DATA57 DATA58 DATA59 DATA60 DATA61 DATA62,
DATA63 DATA64 DATA65 DATA66 DATA67 DATA68 DATA69,
DATA70
CST LOCAL DATA1 DATA2
END
```

The Linkage Editor automatically incorporates the data module UIMDATA for the operation of up to 12 Data Management streams. It makes the name of the program equal to the first four characters of the name of the first code module input. When loaded into the machine the program must be labelled by the external name ENTRYPOINT.

The Linkage Editor produces a map showing the layout of the program and information is placed in the process module to enable run-time symbolic debugging aids to be used.

The automatic linkage features (see the manual OS4000 Linkage Editor Level 2, 85-63463) assume a generalised process structure as supported by the standard user shells of a system. This generalised process structure is not normally applicable when linking processes for use in System Generation or for loading into specialised shells, and in these cases it is better to specify a full set of linkage commands.

CASE

instructs the Linkage Editor to distinguish between upper and lower case which takes effect from the point at which it occurs in the command list, and not otherwise.

CST <name1>[<name 2><name 3><name>]

defines the initial CST, enabling the four segments named to occupy CST0, CST1, CST2 and CST3 respectively when the process starts up for the first time. CST0 holds the local data of the process and its name must be specified in this command. CST1, CST2 and CST3 may be null.

DETAILS <namelist>

defines those segments into which data modules may be placed.

DEBUG

instructs the Linkage Editor to include in the process module a special segment which enables symbolic run-time debugging to be used on the process.

DEFAULT

heads the Linkage Editor's own internal command list and is called into operation when the command stream for a partial set of commands or a null set is exhausted, as these do not contain an END command.

DSEGMENT <name> <number>

enables two code modules within a process that have different local data but the same code to share that code.

END

terminates a full set of commands and causes the linkage to proceed.

ENTRY <name>

defines the entrypoint. The module containing the entry point must be part of the initial segment in CST3. The entry point can be in a library module.

LCOPY <number>

causes the specified number of copies of the local data segment to be placed into successive PAST entries following the PAST entry for the original local data.

LIBRARY <namelist>

specifies those segments into which code modules from the library can be placed. In doing this the Linkage Editor tries to obey the following guidelines:

- Each segment contains a consecutive group of modules based on their input order.
- The distribution of segment sizes is approximately uniform.
- If necessary, to ensure a minimum segment size of approximately 4K bytes, some of the segments are not used.

If these guidelines are not suitable the user can specify the target size of LIBRARY segments.

MAP

instructs the Linkage Editor to produce a map of the process showing its structure and the location of all the named items it contains.

NOLOCAL <namelist>

suppresses the inclusion of local data for any other modules present when showing segments that are used for data or code as, when they contain code, each process must have a copy of the local data for the modules that it uses.

PASLIB <namelist>

defines those segments into which PAS modules may be placed.

PAST <namelist>

equates the segment names on the list to successive PAST numbers beginning at one. PAST entries are not completely interchangeable and therefore the order of the list is important.

PROCESS <name>

defines the process name; only the first four characters of the name are significant.

ROUTES <namelist>

defines route names. The route names on the list are equated to successive route numbers beginning at zero. The function of each route number is pre-defined and therefore the order of the list is important.

SEGMENT <name> [<number>]

may specify a CST number (0,1,2 or 3) in order to fix the virtual address of each module. Each segment command is followed by one or more MODULES commands specifying the modules it contains.

SEGSIZE <number> <number>

Specifies the size of all LIBRARY segments not given individually.

SUPPRESS <namelist>

suppresses the inclusion of segments in the process module when the segment that is to be shared by two or more processes is already in the system, thus enabling the copy in the current process module to be ignored.

VALUE <name> <number>

defines the correspondence between names and values. Routes defined by this means are not checked against the maximum route number defined by the shell into which the process is loaded.

Format

LINK2 FROM stream TO stream WITH stream LIST stream
LIB stream OUT stream WORK stream

Purpose

To link together object modules produced by compilers to form a process module in accordance with a default process linkage description, possibly modified by supplied commands.

Arguments

FROM	Argtype: STREAM 3	Default: $A
	Stream containing the object modules to be linked.	
TO	Argtype: STREAM 4	Default: $B/NEW/LSB
	Stream for process module produced.	
WITH	Argtype: STREAM 5	Default: SINK
	Stream containing commands to modify the default process linkage description.	
LIST	Argtype: STREAM 6	Default: SINK/NEW
	Stream used for Linkage Editor map and error messages.	
LIB	Argtype: STREAM 7	Default: SYS.LINKLIB
	Stream containing the library to be scanned by the Linkage Editor for modules not supplied.	
OUT	Argtype: STREAM 2	Default: $M/NEW
	Stream used for messages.	
WORK	Argtype: STREAM 8	Default:
	Work file.	$W/NEW/LRB16(252)

Messages and Condition Codes

At the end of the linkage:	Condcode
Link complete	0
Link errors	4
Link failure	8

The format of error messages is:

```
*Link error <number> <text> <optional name>
<optional extra information>
```

The error numbers are explained below. <text> is an abbreviated form of the explanation below. Optional information is provided where appropriate. If there are any errors during the command input phase the Linkage Editor will terminate at the end of this phase without attempting to link the process. Errors below which are marked FATAL, also cause immediate termination of the Linkage Editor.

Number	Explanation
1	Fault in the format of an object module. In normal use this implies that the FROM file does not contain object modules (FATAL).
2	The size of the FORTRAN common block is less than expected.
3	The named item is used in two conflicting ways.
4	The named item has been referenced but not defined.
5	The library file does not begin with a correct header record (FATAL).
6	Either the FROM file or the LIB file ends in the middle of an object module (FATAL).
7	Dictionary full (FATAL).
8	Linkage Editor does not continue if there are command errors (FATAL).
9	There is not enough store available to allocate internal workspace (FATAL).
10	Fault in library format. The library length item is absent from the library (FATAL).
11	Reserved.
12	The process contains VS modules so the initial CST1 and CST2 have been nullified. They were not empty however and their contents may not be accessible.
13	The named module is a large VS data module. Either there are not a sufficient number of empty data segments with successive PAST entries to hold the module or it is specified in a MODULES command in conjunction with other modules.
14	Reserved.
15	Reserved.
16	There is not sufficient space to allocate the named module, group of modules or local data.
17	The named module is a data module but the .L facility has been used on it.
18	The entry point of the process is not correctly defined or is not in the initial CST3.
19	The process contains two segments with the same PAST number - that of the named segment.

Number	Explanation
20	The symbol segment is greater than 16384 bytes (FATAL).
21	Followed by a list of mandatory commands which are missing.
22	Current command does not contain a name where expected.
23	Illegally repeated command.
24	Current command does not contain a number where expected.
25	A numeric parameter in the current command is out of range.
26	Illegal combination of commands for specified name.
27	Named segment appears in two SEGMENT commands specifying different CST numbers.
28	The named item appears twice in the same type of command.
29	Incomplete set of commands for the specified name.
30	The current line is not recognised as a command.
31	The current line contains too much information.
32	A library item has been found in the current object module (FATAL).
33	A relocation item does not follow its associated data record (FATAL).
34	Current object module contains no data areas - a module should always contain at least one (FATAL).
35	Current code module contains no code areas - a code module should always contain at least one (FATAL).
36	Reference has been made to a label in a data module (FATAL).
37	In the named module reference has been made to a non-existent area (FATAL).
38	No area exists for the entrypoint of the process (FATAL).
39	An attempt has been made to switch back to the base level module when already at this level (FATAL).
40	End of input has been reached but, due to module switching, structure is not at base level (FATAL).
41	Unexpected end of input (FATAL).
42	Non-symbolic information has been found in the area reserved for symbolic debugging information (FATAL).
43	The length of a data item, in the named object module, exceeds 248 bytes (FATAL).
44	There is not enough room in PASLIB segments for the named PAS module.
45	An insertion of a PAST number has been asked for and the segment in question has a PAST number greater than 255. The LS byte of the PAST number is inserted.

Number	Explanation
46	The entrypoint module has no local data or its local data is not contained in one of the initial CSTs.
47	An attempt has been made to load the named segment, which has a PAST number greater than 255, into an initial CST.
48	The marker byte in the module trailer of the named module indicates an area alignment greater than 4K (FATAL).
49 - 55	Not used in OS4000.

1. 7 Bit code (hexadecimal)
2. With even parity (hexadecimal)
3. Character
4. Card rows punched
5. Coded byte value (hexadecimal)

1	2	3	4	5
00	00	NUL	12-0-9-8-1	B9
01	81	SOH	12-9-1	31
02	82	STX	12-9-2	32
03	03	ETX	12-9-3	33
04	84	EOT	9-7	17
05	05	ENQ	0-9-8-5	9D
06	06	ACK	0-9-8-6	9E
07	87	BEL	0-9-8-7	9F
08	88	BS	11-9-6	56
09	09	HT	12-9-5	35
0A	0A	LF	0-9-5	95
0B	8B	VT	12-9-8-3	3B
0C	0C	FF	12-9-8-4	3C
0D	8D	CR	12-9-8-5	3D
0E	8E	SO	12-9-8-6	3E
0F	0F	SI	12-9-8-7	3F
10	90	DLE	12-11-9-8-1	79
11	11	DC1	11-9-1	51
12	12	DC2	11-9-2	52
13	93	DC3	11-9-3	53
14	14	DC4	9-8-4	1C
15	95	NAK	9-8-5	1D
16	96	SYN	9-2	12
17	17	ETB	0-9-6	96
18	18	CAN	11-9-8	58
19	99	EM	11-9-8-1	59
1A	9A	SUB	9-8-7	1F
1B	1B	ESC	0-9-7	97
1C	9C	FS	11-9-8-4	5C
1D	1D	GS	11-9-8-5	5D
1E	1E	RS	11-9-8-6	5E
1F	9F	US	11-9-8-7	5F
20	A0	space	no punch	00
21	21	!	12-8-7	2F
22	22	"	8-7	0F
23	A3	vary	8-3	0B
24	24	vary	11-8-3	4B
25	A5	%	0-8-4	8C
26	A6	&	12	20
27	27	acute	8-5	0D
28	28	(	12-8-5	2D
29	A9	)	11-8-5	4D
2A	AA	*	11-8-4	4C
2B	2B	+	12-8-6	2E
2C	AC	,	0-8-3	8B
2D	2D	-	11	40
2E	2E	.	12-8-3	2B
2F	AF	/	0-1	81
30	30	0	0	80
31	B1	1	1	01
32	B2	2	2	02
33	33	3	3	03
34	B4	4	4	04
35	35	5	5	05
36	36	6	6	06
37	B7	7	7	07
38	B8	8	8	08
39	39	9	9	10
3A	3A	:	8-2	0A
3B	BB	;	11-8-6	4E
3C	3C	<	12-8-4	2C
3D	BD	=	8-6	0E
3E	BE	>	0-8-6	8E
3F	3F	?	0-8-7	8F

1	2	3	4	5
40	C0	@	8-4	0C
41	41	A	12-1	21
42	42	B	12-2	22
43	C3	C	12-3	23
44	44	D	12-4	24
45	C5	E	12-5	25
46	C6	F	12-6	26
47	47	G	12-7	27
48	48	H	12-8	28
49	C9	I	12-9	30
4A	CA	J	11-1	41
4B	4B	K	11-2	42
4C	CC	L	11-3	43
4D	4D	M	11-4	44
4E	4E	N	11-5	45
4F	CF	O	11-6	46
50	50	P	11-7	47
51	D1	Q	11-8	48
52	D2	R	11-9	50
53	53	S	0-2	82
54	D4	T	0-3	83
55	55	U	0-4	84
56	56	V	0-5	85
57	D7	W	0-6	86
58	D8	X	0-7	87
59	59	Y	0-8	88
5A	5A	Z	0-9	90
5B	DB	[	12-8-2	2A
5C	5C	\	0-8-2	8A
5D	DD	]	11-8-2	4A
5E	DE	up arrow	11-8-7	4F
5F	5F	underline	0-8-5	8D
60	60	grave	8-1	09
61	E1	a	12-0-1	A1
62	E2	b	12-0-2	A2
63	63	c	12-0-3	A3
64	E4	d	12-0-4	A4
65	65	e	12-0-5	A5
66	66	f	12-0-6	A6
67	E7	g	12-0-7	A7
68	E8	h	12-0-8	A8
69	69	i	12-0-9	B0
6A	6A	j	12-11-1	61
6B	EB	k	12-11-2	62
6C	6C	l	12-11-3	63
6D	ED	m	12-11-4	64
6E	EE	n	12-11-5	65
6F	6F	o	12-11-6	66
70	F0	p	12-11-7	67
71	71	q	12-11-8	68
72	72	r	12-11-9	70
73	F3	s	11-0-2	C2
74	74	t	11-0-3	C3
75	F5	u	11-0-4	C4
76	F6	v	11-0-5	C5
77	77	w	11-0-6	C6
78	78	x	11-0-7	C7
79	F9	y	11-0-8	C8
7A	FA	z	11-0-9	D0
7B	7B	{	12-0	A0
7C	FC	|vertical	12-11	60
7D	7D	}	11-0	C0
7E	7E	~overline	11-0-1	C1
7F	FF	DEL	12-9-7	37

BINARY CONVERSION

Dec	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	0001 0000

POWERS OF 16 TABLE

16**n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 848 976	15

POWERS OF 2

2n	n	2-n
2	1	.5
4	2	.25
8	3	.125
16	4	.062 5
32	5	.031 25
64	6	.015 625
128	7	.007 812 5
256	8	.003 906 25
512	9	.001 953 125
1 024	10	.000 976 562 5
2 048	11	.000 488 281 25
4 096	12	.000 244 140 625
8 192	13	.000 122 070 312 5
16 384	14	.000 061 035 156 25
32 768	15	.000 030 517 578 125
65 536	16	.000 015 258 789 062 5
131 072	17	.000 007 629 394 531 25
262 144	18	.000 003 814 697 265 625 5
524 288	19	.000 001 907 348 632 812 25
1 048 576	20	.000 000 953 674 316 406 625
2 097 152	21	.000 000 476 837 158 203 125
4 194 304	22	.000 000 238 418 579 101 562
8 388 608	23	.000 000 119 209 289 550 781 5
16 777 216	24	.000 000 059 604 644 775 390 25
33 554 432	25	.000 000 029 802 322 387 695 313
67 108 864	26	.000 000 014 901 161 193 847 656
134 217 728	27	.000 000 007 450 580 596 923 828
268 435 456	28	.000 000 003 725 290 298 461 914
536 870 912	29	.000 000 001 862 645 149 230 957
1 073 741 824	30	.000 000 000 931 322 574 615 479
2 147 483 648	31	.000 000 000 465 661 287 307 739
4 294 967 296	32	.000 000 000 232 830 643 653 870
8 589 934 592	33	.000 000 000 116 415 321 826 935
17 179 869 184	34	.000 000 000 058 207 660 913 467
34 359 738 368	35	.000 000 000 029 103 830 456 734
68 719 476 736	36	.000 000 000 014 551 915 228 367
137 438 953 472	37	.000 000 000 007 275 957 614 183
274 877 906 944	38	.000 000 000 003 637 978 807 092
549 755 813 888	39	.000 000 000 001 818 989 403 546
1 099 511 627 776	40	.000 000 000 000 909 494 701 773

HEXADECIMAL AND DECIMAL CONVERSION

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	= DEC	HEX	= DEC	HEX	= DEC	HEX	= DEC	HEX	= DEC	HEX	= DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0123		4567		0123		4567		0123		4567	
BYTE				BYTE				BYTE			

85-64732 Issue1

GEC Computers Limited
Elstree Way
Borehamwood
Hertfordshire WD6 1RX
England

tel: 01-953 2030

telex: 22774
reg. no. 712108

Holding Company: The General Electric Company p.l.c. of England