DPS7000/XTA
NOVASCALE 7000

SDI-GPL Primitives

Reference Manual

Languages: General

REFERENCE
47 A2 65UL 06

DPS7000/XTA NOVASCALE 7000

SDI-GPL Primitives

Reference Manual

Languages: General

Software

December 2006

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 65UL 06

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

Preface

Scope and Objectives

This manual describes the GCOS 7 Standard Development Interface (SDI) sets of GPL primitives you need for programmed access:

- to the telecommunications driver (CAM),
- to IOF applications,
- to file and volume management tools,
- to automatic operation facilities.

Each one of these is an optional product, and requires a specific MI (Marketing Identifier) for it to be installed on your system.

Intended Readers

This manual is for use by experienced GPL programmers.

Structure

Chapter 1	*About SDI.* Gives a short description of the SDI product.
Chapter 2	*Telecommunication Manager SDI* Describes the GPL primitives used to access GCOS 7's telecommunication facilities.
Chapter 3	*IOF Application SDI* Describes the GPL primitives used to access GCOS 7's interactive facilities.
Chapter 4	*File Management Tools SDI* Describes the GPL primitives used to access GCOS 7's file and volume facilities.
Chapter 5	*Automatic Operations SDI* Describes the GPL primitives used to access GCOS 7's automatic job management and SYS.LOGC facilities.

Bibliography

GPL System Primitives Reference Manual *47 A2 34UL*
GPL User's Guide *47 A2 36UL*
Data Management Utilities User's Guide *47 A2 34UF*
Catalog Management User's Guide *47 A2 35UF*
TDS GPL Reference Manual *47 A2 27UT*
TDS COBOL Programmer's Guide *47 A2 33UT*
SBR Programmable Interface User's Guide *47 A2 15US*
Introduction to ASM7 ..... *47 A2 48US*
ASM 7 File Migration Administrator's Guide *47 A2 51US*

Syntax Notation

GPL Specific Notation:

The following is a summary of the syntax conventions used to describe the SDI-GPL primitives in this document. For a complete description of GPL syntax and of GPL in general, please refer to the *GPL System Primitives Reference Manual*.

ITEM	An item in upper case is a literal value, to be specified as shown. The upper case is merely a convention; in practice you can specify the item in upper or lower case.
item	An item in lower case is non-literal, indicating that a user-supplied value is expected. An item is usually made up of a prefix, a data type, and a suffix (length).

Prefix:

The prefix indicates whether the item is input, output, or input/output to the primitive.

b_	The item is both input to and output from the primitive.
i_	The item is input to the primitive.
iv_	The item is input to the primitive. It will be used as the argument of an ADDR built-in function, so it must be a variable.
l_	The item is a literal and is input to the primitive.
o_	The item is output from the primitive.

Data Type:

The item also indicates the type of the value.

alphanum	same as char, but restricted to digits (0 through 9), letters (A through Z), hyphen (-), and underscore (_).
bit	string of bits.
char	string of characters.
fb	FIXED BIN.
hex	string of hexadecimal characters, (0 through 9, and A through F).
identifier	same as alphanum, except that it must start with a letter.
structure	structure.
ptr	pointer.

Suffix (Length):

If present, the suffix indicates the length of the value.

n	indicates the maximum length.
_n	indicates the exact length.

EXAMPLES:

i_char	input, character, length = any value.
l_identifier15	input, identifier, length <= 15.
o_char2	output, character, length <= 2.
o_char_20	output, character, length = 20.
i_structure	input, structure.

❑

General Syntax Notation:

ITEM	An underlined item is a default value.
bool	A Boolean value which is either 1 or 0. A Boolean parameter can be specified by its keyword alone, optionally prefixed by "N". Specifying the keyword alone always sets the value to 1. Prefixing the keyword with "N" always sets it to 0.
{ }	Braces indicate a choice of values. Only one may be selected.
[ ]	Square brackets indicate that the enclosed item is optional. An item not enclosed in square brackets is mandatory.
( )	Parentheses indicate that a single value or a list of values can be specified. A list of values must be enclosed by parentheses, with each value separated by a comma or a space.
...	Ellipses indicate that the item concerned can be specified more than once.

Table of Contents

Table of Graphics

Figure

Tables

1. About SDI

The Standard Development Interface (SDI) consists of seven sets of GPL primitives for the advanced use of GCOS 7. These are:

- the Telecommunication Manager SDI (see this manual)
- the IOF Application SDI (see this manual)
- the File Management Tools SDI (see this manual)
- the Automatic Operations SDI (see this manual)
- the TDS Back-End SDI (see the *TDS-COBOL Programmer's Guide*)
- the TDS TPR development in GPL (see the *TDS GPL Reference Manual*)
- the System Behavior Reporter SDI (see the *SBR Programmable Interface User's Guide*)

Each of these SDIs is an optional product which requires a specific MI (Marketing Identifier).

2. Telecommunication Manager SDI

The Telecommunication Manager SDI consists of the following set of advanced GPL primitives:

CAM session management:

H_CACCEPT	accepts a session
H_CATTN	sends a break
H_CDISABLE	closes the mailbox
H_CENABLE	activates the mailbox
H_CGETSEM	allocates a session semaphore
H_CINIT	requests a session
H_CINQ	obtains session attributes
H_CRECEIVE	receives data or an ITA interrupt
H_CREJECT	refuses a session
H_CSEMPOOL	creates a semaphore pool
H_CSEND	sends data
H_CTELEG	sends a telegram
H_CTERM	terminates a session abnormally
H_CTURN	requests the turn
H_DCINQ	declares a structure for the session attributes
H_DCMBX	declares the mailbox structure
H_DCMODEL	declares a list of model identifiers
H_DCMP	declares a message path
H_WAIT	monitors the arrival of a semaphore message

Creation of dynamic operator commands:

H_CRCOM	creates a dynamic operator command
H_DLCOM	deletes a dynamic operator command
H_GETCOM	moves a command to the requestor's area

Task management:

H_NOTIFY	performs a V-operation on a semaphore with message
H_RTNBPROCESS	returns the number of running processes

The GPL syntax conventions are summarized in the preface of this manual. For a complete description of GPL syntax and of GPL in general, please refer to the *GPL System Primitives Reference Manual*.

The Telecommunication Manager SDI set of primitives has a specific MI (Marketing Identifier) which must be validated before program linkage.

2.1 Using CAM

The primitives described in this section provide the interface between a GPL application and the telecommunications driver of the GCOS 7 operating system (CAM). They are used to establish and coordinate a communications session between two GPL applications, with CAM as the intermediary at both ends of the connection. The application that requests the session is called the initiator and the application that receives the request is called the acceptor or destinee; both are called correspondents.

Each correspondent has a unique network address in the form of a mailbox name and a node name. Associated with the mailbox is at least one message path defining the rules under which the correspondent can establish a session. If different types of session are envisaged, the mailbox can have a variety of message paths each suited to a particular type of session.

When an initiator requests a session, it specifies which of its message paths it wants to use. CAM then searches for a compatible message path among those belonging to the acceptor mailbox. If it finds one that is suitable, it transmits the session request to the acceptor, who either accepts or refuses the connection according to its own criteria. If CAM is unable to find a

suitable message path, it rejects the initiator's request and does not notify the acceptor.

Once a session is established, there are two kinds of information that circulate:

- The notifications and acknowledgments that accompany and regulate the normal flow of data between the two correspondents. These are issued according to the dialog rules negotiated at session opening.
- The event-driven interrupts which alter the normal flow of data and precipitate specific types of action. Interrupts override the dialog rules and can be issued at any moment.

Each correspondent chooses its own mode of dialog, either synchronous or asynchronous. The mode is specific to the correspondent and not to the session, and it is quite possible for one correspondent to work in synchronous mode and for the other to work in asynchronous mode. However, all sessions on the same mailbox must be of the same mode.

2.1.1 Message Path Declaration

Message paths are created by the declarative primitive H_DCMP. This defines for each message path:

- the record size that can be sent or received in data transactions,
- which mailbox (initiator or acceptor) has the turn when the session opens,
- various terminal-specific presentation parameters,
- the node and mailbox names of an acceptor mailbox (for use when these are unspecified by the initiator).

One or more of these message paths are attached to the mailbox when the mailbox is activated. When the mailbox requests a session, it specifies which of its attached message paths it wants to use for the session.

2.1.2 Mailbox Activation

Before a correspondent can initiate or accept a session, it must activate its mailbox using the primitive H_CENABLE. This makes the mailbox known to the network and provides CAM the information it needs to connect a session to the mailbox:

- the mailbox name,
- the mailbox semaphore,
- the mailbox request identifier (REQID),
- the mailbox message path(s),
- the maximum number of sessions that can be connected to the mailbox.

Mailbox semaphore

When a mailbox is activated, it is allocated a mailbox semaphore. This is the address where the mailbox receives the semaphore-driven messages to open or close a session (OPENREQ or SHUTDOWN). If the mode is asynchronous and if there is no other semaphore specific to the session, it is also used for the dialog management messages (OPENACK, CREDIT, DATA, ...).

The same semaphore can be shared between more than one mailbox. Thus H_CENABLE can specify a semaphore that is already allocated to another mailbox through a previous H_CENABLE.

Request Identifier (REQID)

The REQID specified at mailbox activation accompanies the semaphore messages OPENREQ and SHUTDOWN when they arrive at the mailbox semaphore. The REQID thus enables the correspondent to identify the source of these messages.

The same REQID is returned with dialog-related semaphore messages (DATA, CREDIT, INTERUPT) and the OPENACK message if REQID is not redefined at session opening.

2.1.3 Session Initiation

A correspondent can request connection to another correspondent using the primitive H_CINIT. The initiator mailbox must be active and, for the request to be received, the destinee mailbox must also be active.

Using H_CINIT, the initiator specifies:

- which of its message paths it wants to use,
- the node and mailbox name of the destinee (if different to that declared by H_DCMP),
- the dialog mode (synchronous or asynchronous),
- a session semaphore and request identifier (if mode is asynchronous),
- user, billing, project, station, and password details,
- details specific to an IOF or TDS-type session.

In return, CAM generates a session reference (SREF).

Synchronous mode:

The correspondent receives a return code indicating whether the session has been accepted or rejected by the destinee.

Asynchronous mode:

The correspondent receives a return code indicating whether the request has been transmitted to the destinee. If the return code is DONE, the initiator then executes H_WAIT to receive the destinee's OPENACK response (at the session semaphore if this is specified, otherwise at the mailbox semaphore).

If the OPENACK response contains BYTE1=0, the session is accepted. If BYTE1 <> 0, the session is refused and the reason is given by the value returned by BYTE2 and BYTE3.

In addition to the OPENACK response, the initiator receives the REQID it specified at mailbox activation or at session initiation. This enables the initiator engaged in multi-session dialog to identify the source of the OPENACK response.

After it receives the OPENACK with BYTE1=0 response, the initiator must execute the H_CINQ primitive in order to obtain the dialog management options negotiated for the session.

2.1.4 Session Negotiation

When a correspondent initiates a session it specifies explicitly which of its mailbox message paths it wants to use for the session. CAM compares this message path with each message path attached to the acceptor mailbox. This operation is called parameter negotiation. Its aim is to determine which dialog management parameters are common to both mailboxes.

The first message path which satisfies all the negotiation rules is selected. If none of the message paths is suitable, the acceptor is not notified of the request and the initiator receives a rejection message.

If parameter negotiation is successful, CAM transmits the OPENREQ message to the acceptor who must then execute the H_CINQ to obtain the negotiated parameters. If these are acceptable, it can execute the H_CACCEPT primitive to send the OPENACK BYTE1=0 response to the initiator. When the initiator receives the OPENACK BYTE1=0 response it too must execute H_CINQ to obtain the negotiated parameters.

Some message path parameters are negotiable, some are not. The non-negotiable parameters are simply transported across to the other end of the session and retrieved, along with the negotiated parameters, by the H_CINQ primitive.

The negotiable parameters are the following:

MODE	negotiated under Rule 1.
IRECSZ	negotiated under Rule 2.
ORECSZ	negotiated under Rule 2.
EXTDATN	negotiated under Rule 1b.
DVHDR	negotiated under Rule 3.
CODE	negotiated under Rule 1.
SYMBOLS	negotiated under Rule 1.

Note that IRECSZ on one side of the session is negotiated with ORECSZ on the side, and vice versa.

2.1.4.1 Rule 1

Each correspondent proposes a set of possible values for the negotiable parameter; the intersection of these two sets determines the common value.

- If the intersection is empty:

  rule 1 => the negotiation fails
  rule 1b => the negotiation succeeds

- If the intersection contains more than one value, the first in the list applies.

2.1.4.2 Rule 2

Each correspondent proposes a maximum value; the common value is the smaller of the two.

2.1.4.3 Rule 3

Each correspondent describes the option in one of the following ways:

11 - mandatory
01 - optional
00 - unsupported

- The option is *retained* if it is:
  - mandatory for both correspondents
  - optional for both correspondents
  - mandatory for one and optional for the other
- The option is *not retained* if it is:
  - unsupported by both correspondents
  - unsupported by one correspondent and optional for the other
- The negotiation *fails* if the option is:
  - unsupported by one correspondent and mandatory for the other

The above permutations are summarized in the following table.

Table 2-1. Negotiation Rule 3

	INITIATOR		
ACCEPTOR	00	01	11
00	not retained	not retained	FAILURE
01	not retained	retained	retained
11	FAILURE	retained	retained

2.1.5 Session Acceptance

Once its mailbox is activated, a correspondent can issue the H_WAIT primitive and then wait for a session opening request (OPENREQ) to arrive at its mailbox semaphore.

The OPENREQ message, when it arrives, is accompanied by the request identifier (REQID) specified at mailbox activation. The acceptor also receives the session reference (SREF) generated by CAM at session initiation.

After receiving an OPENREQ message, the acceptor must execute the H_CINQ primitive in order to obtain the message path options negotiated for the session and the user/project/billing identity of the initiator. The reception zone for this information is defined by the declarative primitive H_DCINQ.

Having executed H_CINQ, the acceptor can either accept or refuse the session. It accepts by executing H_CACCEPT and supplying the following information:

- the session reference,
- whether it wants to dialog in synchronous or asynchronous mode,
- a REQID for the current session, if engaged in multi-session dialog,
- some IOF or TDS-specific options.

The session can be refused, and the reason for the refusal returned, by executing the H_CREJECT primitive.

2.1.6 Session Dialog

Once a session between two correspondents is established (requested, accepted, and the dialog parameters obtained), the dialog can begin. The primitives H_CSEND and H_CRECEIVE are used to send and receive data.

Each correspondent dialogs with their local CAM interface, and it is quite possible for the dialog to be in synchronous mode at one end of the session and in asynchronous mode at the other end. However, all sessions open on the same mailbox must be of the same mode.

Asynchronous dialog

This form of dialog is semaphore-driven and allows the correspondent to await events (CREDIT, DATA, timer, OMH, ...) and at the same time continue processing other events.

Synchronous dialog

This form of dialog is driven by CAM-generated return codes which follow the execution of each primitive. The correspondent must await the return code of the current primitive before proceeding with the next.

2.1.6.1 Sending Data

The H_CSEND primitive is used to send a data record, an end-of-interactions, or an end-of-session. It must specify:

- the session reference,
- the zone containing the data to be sent,
- the length of the data to be sent,
- the level of the data:

LEVEL=1	=> a data record
LEVEL=3	=> an end-of-interaction
LEVEL=5	=> an end-of-session

- a zone for receiving the INTERUPT return code (if the send fails).

The rules concerning what primitives must precede or follow an H_CSEND depend on the correspondent's dialog mode: synchronous, asynchronous TWA, or asynchronous TWS.

Synchronous mode

To send data in synchronous mode, the correspondent must:

- get the turn,
- issue an H_CSEND.

The H_CSEND is executed as soon as there is sufficient buffer space for the data to be sent, but until then, no other primitive can be executed.

Parameter negotiation determines which correspondent has the turn first. Thereafter, the correspondent gets the turn when it issues an H_CRECEIVE and obtains an end-of-interaction (LEVEL=3) from its partner.

The correspondent keeps or loses the turn depending on the data level it specifies with each H_CSEND:

LEVEL=1	=> send a data record, and keep the turn
LEVEL=3	=> send an end-of-interaction, and lose the turn
LEVEL=5	=> send an end-of-session, and lose the turn permanently

After a LEVEL=3 send, the correspondent cannot proceed with its next primitive until the partner executes an H_CRECEIVE primitive to receive the

data. Nor can it execute another H_CSEND until it gets the turn back; that is, not until it receives a LEVEL=3 send by its partner. See "Receiving Data" below.

Asynchronous TWA mode

To send data in asynchronous TWA mode, the correspondent must:

- get the turn
- issue an H_WAIT to monitor the arrival of the CREDIT semaphore message
- issue an H_CSEND

The H_CSEND is executed when the CREDIT semaphore message arrives (a signal that there is sufficient buffer space for the data to be sent).

Parameter negotiation determines which correspondent has the turn first. Thereafter, the correspondent gets the turn when it issues an H_CRECEIVE and obtains an end-of-interaction (LEVEL=3) from the partner.

The correspondent keeps or loses the turn depending on the data level it specifies with each H_CSEND:

LEVEL=1	=> send a data record, and keep the turn
LEVEL=3	=> send an end-of-interaction, and lose the turn
LEVEL=5	=> send an end-of-session, and lose the turn permanently

After a LEVEL=3 send, the correspondent must issue an H_WAIT to monitor the arrival of the DATA semaphore message. Its arrival indicates that the partner has received the data, and is either sending data of its own or relinquishing the turn. To receive the data (or the turn), the correspondent must follow the H_WAIT with an H_CRECEIVE. See "Receiving Data" below.

Asynchronous TWS mode

The "turn" concept does not exist. To send data in asynchronous TWS mode, the correspondent must:

- issue an H_WAIT to monitor the arrival of the CREDIT semaphore message
- issue an H_CSEND

The H_CSEND is executed when the CREDIT semaphore message arrives (a signal that there is sufficient buffer space for the data to be sent).

In TWS mode, there is no distinction between LEVEL=1 and LEVEL=3:

LEVEL=1	=> send a data record
LEVEL=3	=> send a data record
LEVEL=5	=> send an end-of-session

After executing a LEVEL=1 or LEVEL=3 send, the correspondent must issue an H_WAIT to monitor the arrival of either a CREDIT or a DATA semaphore message. If CREDIT, it issues another send; if DATA, it issues a receive.

H_CSEND interrupt

If execution of the H_CSEND primitive generates the return code INTERUPT, then:

- the data is not sent; it remains in its transmission zone
- the type of interrupt (ATTENTION, TERMREQ, ...) is given by the ITA zone

See "Interrupts" below for details.

2.1.6.2 Receiving Data

The H_CRECEIVE primitive is used to receive a data record, an end-of-interaction, or an end-of-session. It must specify:

- the session reference
- the zone for receiving the data
- the length of the data reception zone
- the zone for receiving the data length
- the zone for receiving the data level:

LEVEL=1	=> data record
LEVEL=3	=> end-of-interaction (get the turn)
LEVEL=5	=> end-of-session

- the zone for receiving the INTERUPT return code (if the receive fails)

The rules concerning what primitives precede or follow an H_CRECEIVE depend on the correspondent's dialog mode: synchronous, asynchronous TWA, or asynchronous TWS.

Synchronous mode

To receive data in synchronous mode, the correspondent must:

- relinquish the turn with a LEVEL=3 send
- issue an H_CRECEIVE at regular intervals until it gets the turn back

The H_CRECEIVE is executed if there is data to be received, otherwise it is ignored.

The correspondent regains the turn if the H_CRECEIVE returns LEVEL=3 or LEVEL=5.

Asynchronous TWA mode

To receive data in asynchronous TWA mode, the correspondent must:

- relinquish the turn with a LEVEL=3 send,
- issue an H_WAIT to monitor the arrival of the DATA semaphore message,
- issue an H_CRECEIVE.

The H_CRECEIVE is executed when the DATA semaphore message arrives (a signal that data has been sent).

The correspondent regains the turn if H_CRECEIVE returns LEVEL=3 or LEVEL=5. If LEVEL=1, the correspondent must reissue an H_WAIT followed by an H_CRECEIVE to receive the next send.

Asynchronous TWS mode

The "turn" concept does not exist. To receive data in asynchronous TWS mode, the correspondent must:

- issue an H_WAIT to monitor the arrival of the DATA semaphore message,
- issue an H_CRECEIVE.

The H_CRECEIVE is executed when the DATA semaphore message arrives (a signal that data has been sent).

If H_RECEIVE returns LEVEL=1 (or 3), the correspondent must reissue an H_WAIT to monitor the arrival of either a CREDIT or a DATA semaphore message. If CREDIT, it issues a send; if DATA, it issues another receive.

H_CRECEIVE interrupt

If execution of the H_CRECEIVE primitive generates the return code INTERUPT, then:

- the data being received is not delivered,
- the type of interrupt (ATTENTION, TERMREQ, ...) is given by the ITA zone.

See "Interrupts" below for details.

2.1.6.3 Semaphore Messages

There are seven semaphore messages returned by H_WAIT. The semaphore which is to wait for these messages is defined by the correspondent at mailbox activation (H_CENABLE) and at session opening (H_CINIT or H_CACCEPT).

OPENREQ	The correspondent is asked to accept a session. It must issue H_CINQ to obtain the dialog management parameters negotiated for the session, and then H_CACCEPT to accept or H_CREJECT to refuse the session.
OPENACK	The correspondent is notified that its request to open a session has been accepted (BYTE1=0) or refused (BYTE1<>0). If accepted, the correspondent must issue H_CINQ to obtain the dialog management parameters negotiated for the session.
DATA	The correspondent is notified that data is waiting to be received; it must execute H_CRECEIVE.
CREDIT	The correspondent is notified that there is sufficient buffer space for its data to be sent; an H_CSEND can be executed.
MGCLOSED	The correspondent is notified that the session is (abnormally) terminated.
SHUTDOWN	The correspondent is notified that the telecommunication supervisor through which the

	session is established, has been terminated. The correspondent must issue H_CDISABLE to deactivate its mailbox.
INTERUPT	The correspondent is notified that an interrupt has arrived; an H_CRECEIVE must be executed to receive the type of interrupt in the ITA zone.

2.1.6.4 REQID Principles

A correspondent can receive semaphore messages from outside the session (timer, OMH, ...), as well as those from CAM (DATA, CREDIT, ...). With each semaphore message, the correspondent receives a REQID which enables it to identify the source of the message (CAM, timer, OMH, ...).

The REQID returned is either that specified at mailbox activation (H_CENABLE) or that specified at session opening (H_CINIT or H_CACCEPT).

In the case of the OPENREQ and SHUTDOWN messages, the REQID is always that specified at mailbox activation, never that specified at session opening.

In all other cases (OPENACK, DATA, CREDIT, INTERUPT, timer messages, OMH messages, ...), the REQID is always that specified at session opening. In multi-session dialog, a REQID must be specified explicitly for each session. In single-session dialog, a REQID can be specified explicitly or it can default to that specified at mailbox activation.

2.1.6.5 Interrupts

A correspondent is notified of an interrupts in two ways:

- by the INTERUPT semaphore message returned by H_WAIT, in which case the correspondent must execute H_CRECEIVE in order to receive the type of interrupt in the ITA zone,
- by the INTERUPT return code when executing an H_CSEND, H_CRECEIVE, H_CTURN, H_CATTN, or H_CTELEG primitive, in which case the type of interrupt is given directly in the ITA zone.

In every case, the primitive being called is not executed, and its entry parameters and the "turn" remain unchanged.

Interrupt Type

There are five types of interrupt:

(0) ATTENTION	A break has arrived (H_CATTN has been executed).
(1) DMNDTURN	A request for the turn has arrived (H_CTURN has been executed).
(3) ABNTERM	The session has been abnormally terminated by one of the following: – execution of the H_CTERM primitive – an irrecoverable line error – execution of the operator command TTSVR STRONG – a $*$ DIS command by the terminal user
(8) TERMREQ	The telecommunications supervisor (for the session) has been terminated by the operator command TTSVR WEAK.
(9) TELEG	A telegram has arrived (sent by H_CTELEG).

Interrupt format

Apart from the TELEG interrupt, each interrupt received is 3 bytes long:

- 1 byte for the interrupt type,
- 2 bytes for the interrupt code.

The TELEG interrupt can be up to 145 bytes long:

- 1 byte for the interrupt type,
- 2 bytes for the interrupt code,
- 2 bytes for the telegram length,
- 140 bytes for the telegram data.

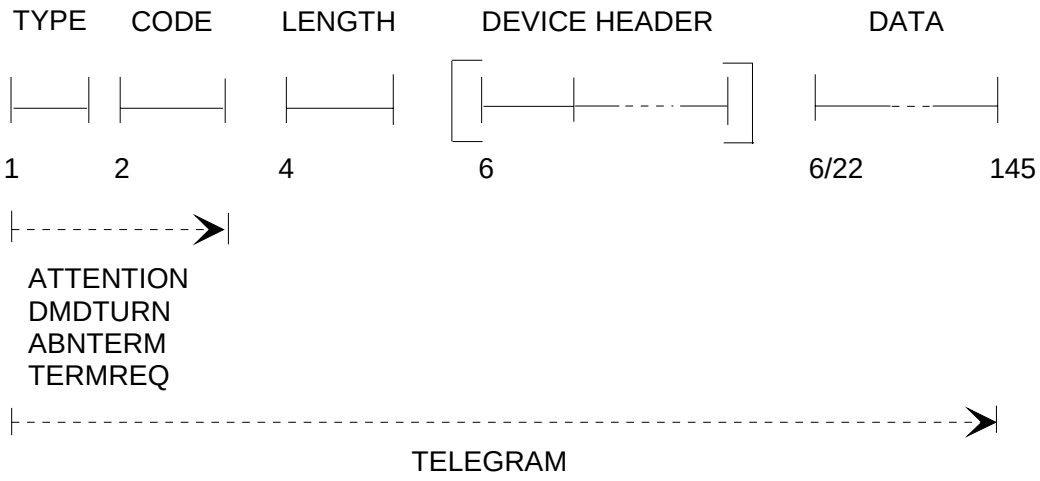


Figure 2-1. Format of the ITA Zone

Interrupt Code

The interrupt code is specified via the parameter ITCODE by the correspondent sending the interrupt (default value is "0000"X).

For telegrams, the interrupt code is at "0000"X.

Interrupt length

Applicable to telegram-type interrupts only. This is the length in bytes of the telegram data. It does not include the length of the DEVICE HEADER.

Data

Applicable to telegram-type interrupts only. This is the data transmitted from the WA zone by H_CTELEG.

If the DEVICE HEADER option has been negotiated successfully, this is inserted, with its length, in between the interrupt's length and data zones.

2.1.7 Session Termination

A session can be terminated either normally by a LEVEL=5 send, or abnormally by the H_CTERM primitive.

2.1.7.1 Normal Termination

Normal session termination begins when either one of the correspondents issues an H_CSEND with LEVEL=5; the termination is completed when the other correspondent also issues an H_CSEND with LEVEL=5.

The initiator of the termination can continue to receive data, but can no longer send data after it executes a LEVEL=5 send. The session terminates as soon as it receives a LEVEL=5 send from its partner.

2.1.7.2 Abnormal Termination

A correspondent can issue the H_CTERM primitive to (abnormally) terminate a session at any moment except between the arrival of the OPENREQ message and execution of the H_CACCEPT primitive.

If, in asynchronous mode, a correspondent issues H_CTERM before the session is accepted (that is, in between H_CINIT and the arrival of the OPENACK response), it receives the MGCLOSED semaphore message. The acceptor is notified of the interrupt on execution of its H_CACCEPT.

Asynchronous mode

When a correspondent executes H_CTERM, the destinee is notified:

- either by the INTERUPT semaphore message returned by H_WAIT, in which case it must execute H_CRECEIVE in order to receive ITA=ABNTERM,
- or by the INTERUPT return code (primitives H_CSEND, H_CRECEIVE, H_CATTN, H_CTELEG, or H_CTURN), in which case it receives ITA=ABNTERM directly.

In either case, the destinee must issue an H_WAIT in order to monitor the arrival of the MGCLOSED semaphore message. When this arrives, the session is terminated. The initiator too must issue an H_WAIT to await the MGCLOSED message.

Synchronous mode

When a correspondent executes H_CTERM, the destinee is notified by the INTERUPT return code with ITA=ABNTERM when it issues an H_CSEND, H_CRECEIVE, H_CATTN, H_CTELEG, or H_CTURN. The session terminates at this point for the destinee, and at H_CTERM's return code for the initiator.

2.1.7.3 Termination by Operator Command (TTSVR)

The telecommunications supervisor may be terminated by the operator command TTSVR when there are sessions still active.

- If TTSVR OPTION=WEAK, all correspondents concerned are notified by the interrupt TERMREQ. Each correspondent must then terminate their session, and no new sessions can be opened.
- If TTSVR OPTION=STRONG, all correspondents concerned are notified by the interrupt ABNTERM.

If all telecommunication supervisors are terminated, then each active correspondent receives the SHUTDOWN message at its mailbox semaphore. This indicates that no further external communication is possible, either with a remote node or with a front-end processor. It is preferable in this case, to terminate internal sessions (those which do not use the network) with a standard or CAM-type local application. An active mailbox prevents an NG network generation.

2.1.8 Mailbox Deactivation

Once all sessions connected to the mailbox have been terminated, the correspondent must issue the H_CDISABLE primitive to deactivate the mailbox.

2.1.9 Program linkage

Non-TDS applications which interface with CAM must specify LINKTYPE=VCAM at program linkage.

2.1.10 Example of Asynchronous (TWA) Dialog

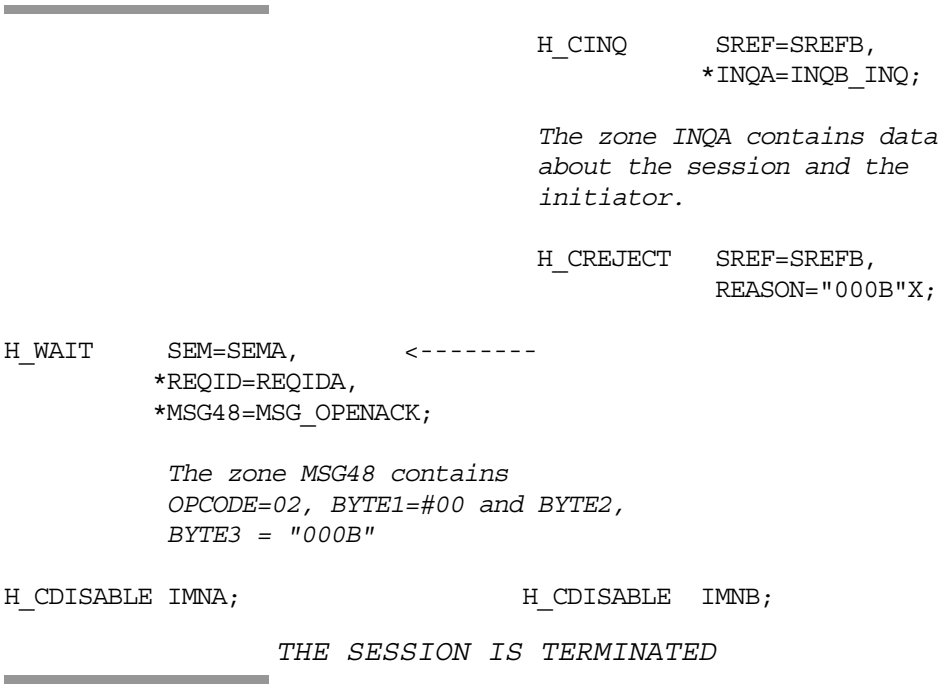
The following example shows the exchanges between two correspondents, both dialoging in asynchronous TWA mode. The example only demonstrates the use of CAM primitives; it is not an example of a complete GPL program.

All lines preceded by "*" represent parameter values returned by CAM.

2.1.10.1 Session Requested

```
H_CENABLE   IMNA, MBX="MBXA",          H_CENABLE   IMNB,MBX="MBXB",
           *SEM=SEMA,                              *SEM=SEMB,
            REQID=REQIDA;                           REQID=REQIDB,
                                                    MPLIST=(IPNB_MP);

H_CINIT     IMNA, MP=IPNA_MP,
           *SREF=SREFA,
            DESTNODE=" ",
            DESTMBX="MBXB";
                    ---------->    H_WAIT      SEM=SEMB,
                                               *REQID=REQIDB,
                                               *MSG48=MSG_OPENREQ;

                                   The zone MSG48 contains
                                   OPCODE=01 and the SREF
```

2.1.10.2 Session Refused



2.1.10.3 Session Accepted

```
H_CINQ      SREF=SREFB,
                                                *INQA=INQB_INQ;

                                    The zone INQA contains data
                                    about the session and the
                                    initiator.

                                    H_CACCEPT   SREF=SREFB,
                                                ASYNC;
H_WAIT      SEM=SEMA,       <--------
           *REQID=REQIDA,
           *MSG48=MSG_OPENACK;

           The zone MSG48 contains
           OPCODE=02, BYTE1=00

H_CINQ      SREF=SREFA
           *INQA=INQA_INQ;

                   SESSION DIALOG CAN BEGIN
```

2.1.10.4 Session Dialog

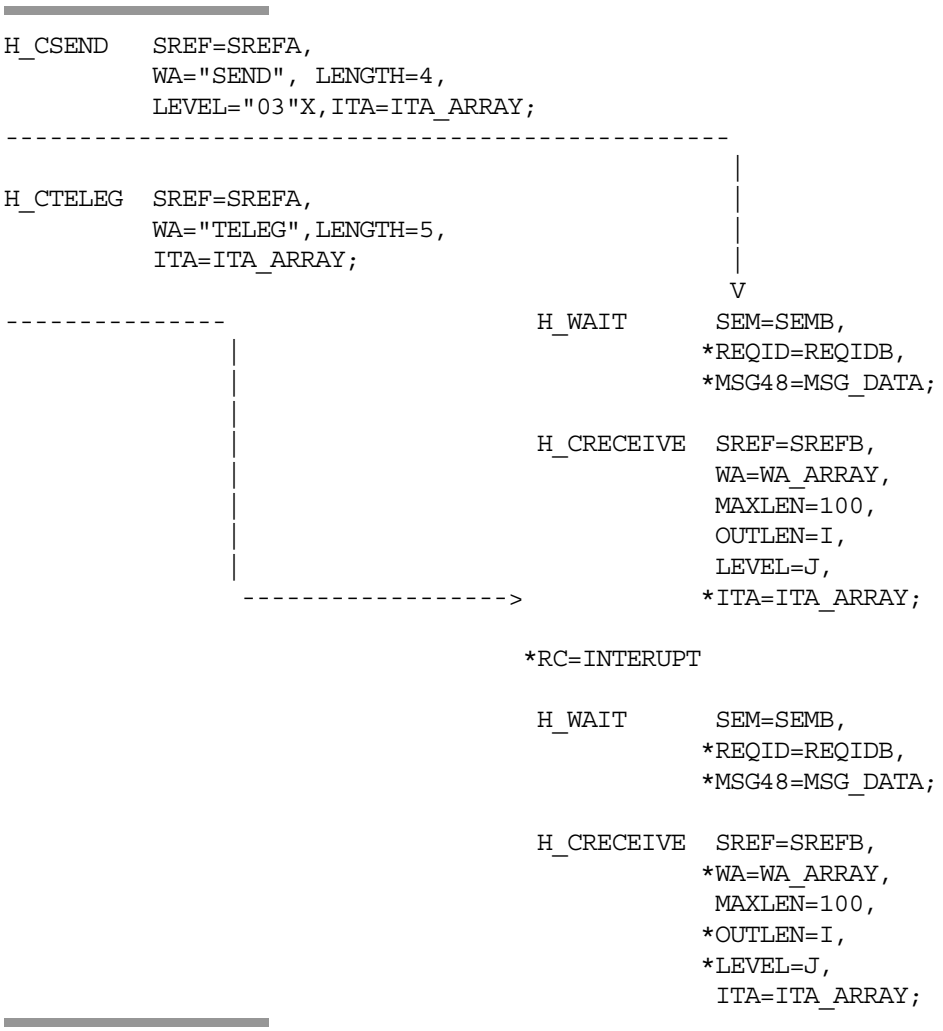
```
H_WAIT   SEM=SEMA,
        *REQID=REQIDA,
        *MSG48=MSG_CREDIT;

         The zone MSG48 contains
         OPCODE = 05

H_CSEND  SREF=SREFA,
         WA="SEND", LENGTH=4,
         LEVEL="01"X,
         ITA=ITA_ARRAY;

---------------------------------->
                                    H_WAIT       SEM=SEMB,
                                                *REQID=REQIDB,
                                                *MSG48=MSG_DATA;

                                    The zone MSG48 contains
                                    OPCODE=04 and BYTE1=01 (LEVEL)

                                    H_CRECEIVE   SREF=SREFB,
                                                *WA=WA_ARRAY,
                                                 MAXLEN=100,
                                                *OUTLEN=I,
                                                *LEVEL=J,
                                                 ITA=ITA_ARRAY;
<----------------------------------

H_WAIT    SEM=SEMA,
         *REQID=REQIDA,
         *MSG48=MSG_CREDIT;
H_CSEND   SREF=SREFA,
          WA="SEND", LENGTH=4,
          LEVEL="03"X,
          ITA=ITA_ARRAY;

---------------------------------->
                                    H_WAIT      SEM=SEMB,
                                               *REQID=REQIDB,
                                               *MSG48=MSG_DATA;

                                    H_CRECEIVE  SREF=SREFB,
                                               *WA=WA_ARRAY,
                                                MAXLEN=100,
                                               *OUTLEN=I,
                                               *LEVEL=J,
                                                ITA=ITA_ARRAY;

                                    H_WAIT      SEM=SEMB,
                                               *REQID=REQIDB,
                                               *MSG48=MSG_CREDIT;
```

2.1.10.5 Interrupts

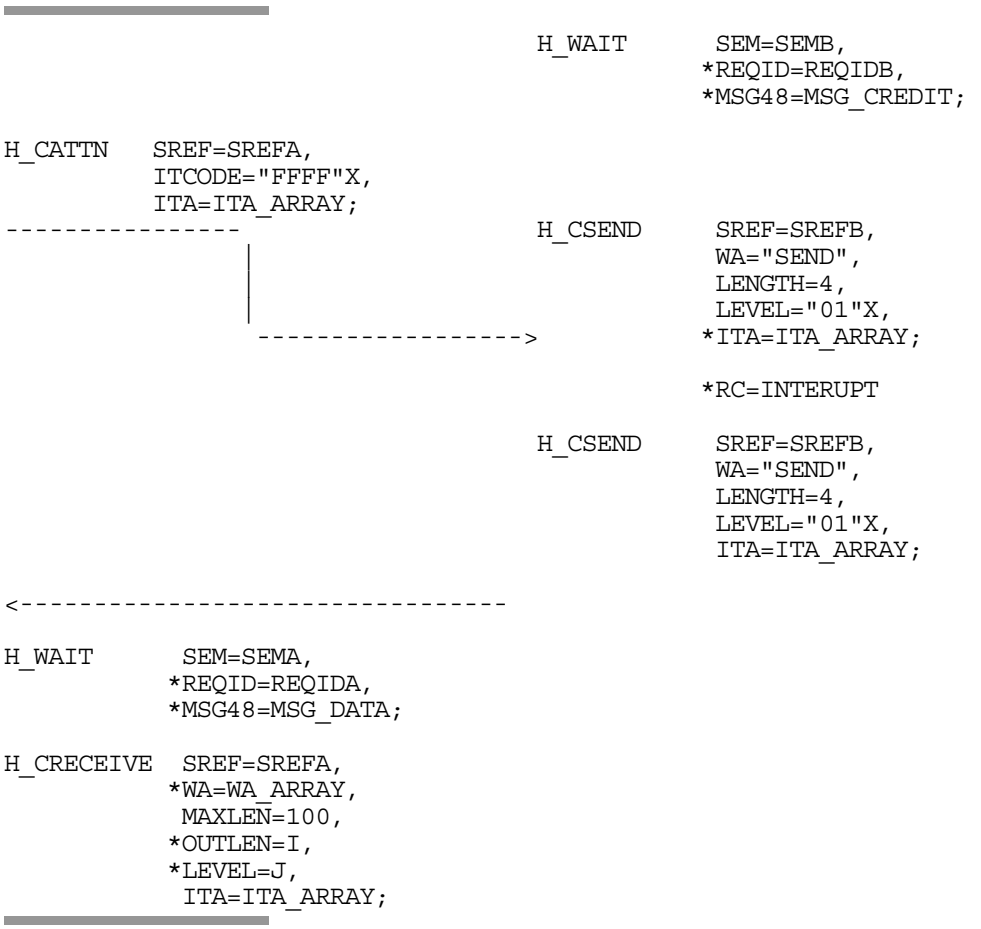
DMDTURN (received on H_WAIT)

```
H_CTURN    SREF=SREFA,
           ITCODE="0010"X,
           ITA=ITA_ARRAY;

---------------------------------->
                                     H_WAIT       SEM=SEMB,
                                                 *REQID=REQIDB,
                                                 *MSG48=MSG_IT;

                                     H_CRECEIVE   SREF=SREFB,
                                                  WA=WA_ARRAY,
                                                  MAXLEN=100,
                                                  OUTLEN=I,
                                                  LEVEL=J,
                                                 *ITA=ITA_ARRAY;
                                                 *RC=INTERUPT
<----------------------------------
H_WAIT     SEM=SEMA,
          *REQID=REQIDA,
          *MSG48=MSG_CREDIT;

           The zone MSG48 contains OPCODE=05

H_CSEND    SREF=SREFA,
           WA="SEND", LENGTH=4,
           LEVEL="01"X,
           ITA=ITA_ARRAY;

---------------------------------->
                                     H_WAIT       SEM=SEMB,
                                                 *REQID=REQIDB,
                                                 *MSG48=MSG_DATA;

                                     H_CRECEIVE   SREF=SREFB,
                                                 *WA=WA_ARRAY,
                                                  MAXLEN=100,
                                                 *OUTLEN=I,
                                                 *LEVEL=J,
                                                  ITA=ITA_ARRAY;

<----------------------------------

H_WAIT     SEM=SEMA,
          *REQID=REQIDA,
          *MSG48=MSG_CREDIT;
```

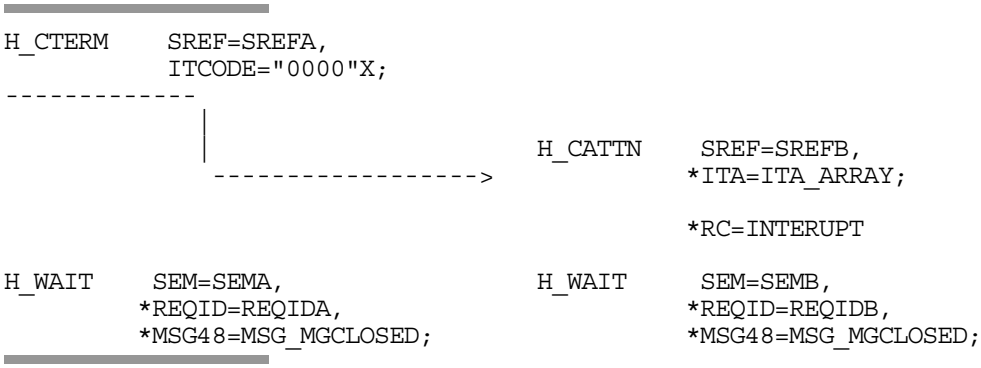
TELEGRAM (received on H_CRECEIVE)



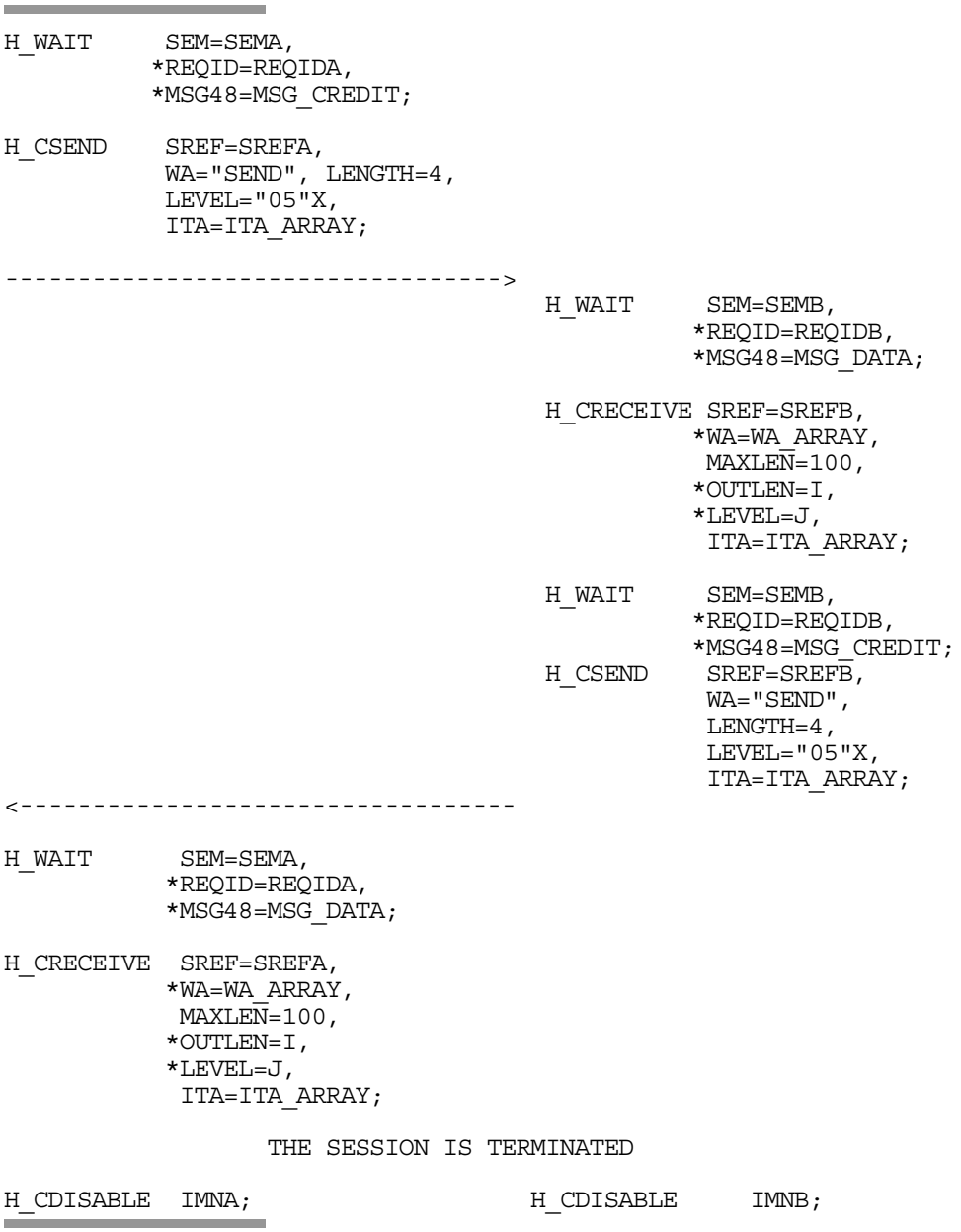
ATTENTION (received on H_CSEND)



ABNTERM (received on H_CATTN)



2.1.10.6 Session Termination (Normal)



2.1.11 ARGERR Return Codes

This subsection explains the 'ARGERR' return codes of the following type:

```
85 XX 1853 --> CA  DD,ARGERR
```

where XX is the code's hexadecimal value and DD its decimal value.

CAM Memory Management

01 Invalid mailbox reference (< 1 or > Max Mailbox)

02 Mailbox control block not found

03 Invalid session reference (< 1 or > Max Session)

04 Session control block not found

05 Invalid semaphore reference (< 1 or > Max Sem)

06 Semaphore control block not found

07 Segment creation not possible (H_SGCR)

08 Semaphore allocation not possible (H_SSMGET)

09 Cannot set the timer (H_SETPTM)

0A Segment deallocation not possible (H_SGDL)

0B Invalid type (internal error)

0C Semaphore liberation not possible (H_SSMFRE)

H_CACCEPT

19 Message path parameter ENTRYNB is invalid

1B Session not active

0D H_CINQ not executed

0E Session acceptance not possible

01 to 0C
See the return codes for CAM memory management

2A to 2F
See the return codes for H_CSEMPOOL

H_CATTN

0F Session not active

01 to 0C
See the return codes for CAM memory management

H_CDISABLE

10 and 11
Mailbox cannot be deactivated

01 to 0C
See the return codes for CAM memory management

H_CENABLE

12 Invalid SEM parameter

13 Parameter MAXS greater than the total number of sessions possible

14 Message path parameter ENTRYNB invalid

15 and 16
Mailbox cannot be activated

17 Message path cannot be activated

01 to 0C
See the return codes for CAM memory management

2A to 2F
See the return codes for H_CSEMPOOL

H_CINIT

1C Cannot retrieve user identity

1D Session cannot be initialized. Local mailbox semaphore invalid

1E Message path parameter ENTRYNB invalid

01 to 0C
See the return codes for CAM memory management

2A to 2F
See the return codes for H_CSEMPOOL

H_CINQ

23 Session not active or H_CINQ already called

24 Cannot execute H_CINQ

01 to 0C
See the return codes for CAM memory management

H_CRECEIVE

25 Session not active

26 H_CINQ not executed: invalid WA or LENGTH parameter

27 Data reception not possible

00 Interrupt reception not possible

01 to 0C
See the return codes for CAM memory management

H_CREJECT

28 Session is not active

29 Session rejection not possible

01 to 0C
See the return codes for CAM memory management

H_CSEMPOOL

2A Parameters NBSEM and/or MNMESS invalid

2B Cannot create a semaphore pool

2C Cannot allocate a semaphore

2D Cannot liberate a semaphore

2E Semaphore not found

2F Invalid TYPE function

H_CSEND

30 Invalid LEVEL parameter

31 Invalid TYPE parameter

32 Session not active

33 H_CINQ not executed: invalid WA or LENGTH parameter

34 Conflict between parameters LEVEL and TYPE

35 Cannot send the data, the CREDIT message not obtained

36 Cannot send the interrupt

01 to 0C
See the return codes for CAM memory management

H_CTELEG

37 Session not active

01 to 0C
See the return codes for CAM memory management

H_CTERM

38 Session not active

39 Session termination not possible

01 to 0C
See the return codes for CAM memory management

H_CTURN

3A Session not active

01 to 0C
See the return codes for CAM memory management

H_WAIT

3B Invalid SEM parameter

3C Invalid status after receiving an event

3D Invalid status after internal reception of data

3E Returned REQID unknown to CAM

3F Returned OPCODE not expected by CAM

01 to 0C
See the return codes for CAM memory management

2.1.12 Device Header

The Device Header (HDR) is a string of 1 to 17 characters used in dialog with:

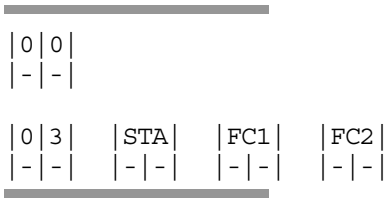
- VIP synchronous terminals
- the 3270 and 2780 series of terminal

It transmits information concerning the terminal's characteristics.

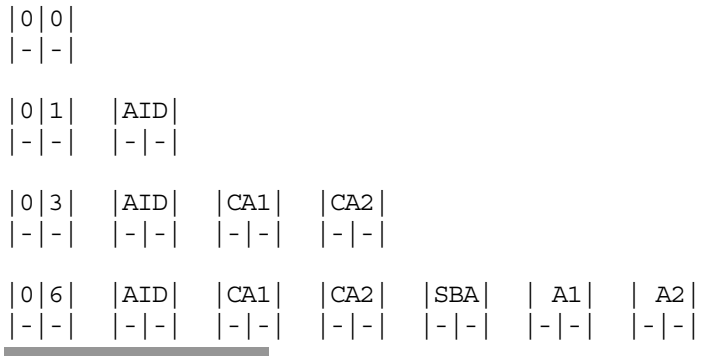
Format

The first byte of the HDR zone gives, in binary form, the number of characters in the Device Header, as follows:

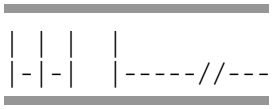
VIP format:



3270 family:



2780 family:



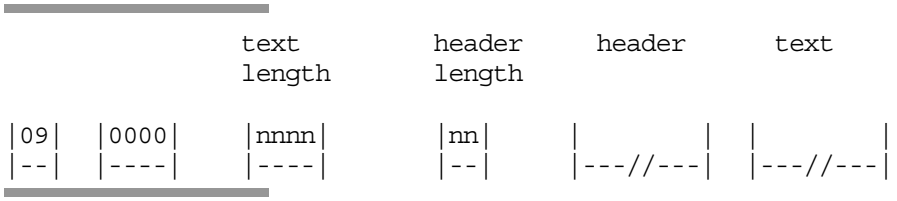
For this transmission, the GPL format for the device header is as follows:

```
DCL 1   HDR,
      2  BSC 1    BIT(1),
      2  BSC 2    BIT(1),
      2   *       BIT(2),
      2  LENGTH   BIT(4),
      2  TEXT     CHAR(15);
```

Programming

If a correspondent requires the device header option, it must specify DVHDR in its message path. The option is negotiated (see subsection "Session Negotiation") with that of its partner correspondent.

If successfully negotiated, the dialog primitives (H_CSEND, H_CRECEIVE, and H_CTELEG) must specify the HDR parameter. On reception of telegram-type interrupts, the device header is returned in the ITA zone with the following format:



If the parameter HDR is not specified (and the device header option has been negotiated), then:

- for transmission, CAM generates a device header at "00"X
- for reception, the device header is lost, except in the case of telegram reception

The correspondent must always take into account the device header length in order to keep within the negotiated record sizes (IRECSZ and ORECSZ returned by H_CINQ).

2.1.13 Limits of Usage

The maximum number of sessions that can be open simultaneously is 2048

The maximum number of mailboxes that can be active simultaneously is 240

2.2 CAM Primitives

2.2.1 H_CACCEPT

Function

Accepts a session.

Syntax

```
$H_CACCEPT
          SREF = i_fb15
          [{,ASYNC [,SEM =b_ptr] [,REQID = i_bit_16]] | ,SYNC }]
          [,OPTION = i_char_23];
```

Parameters

SREF	Session reference returned by CAM with semaphore message OPENREQ
ASYNC	Dialog between acceptor and CAM is to be asynchronous; mandatory in TWS mode.
SYNC	Dialog between acceptor and CAM is to be synchronous
SEM	Semaphore used to manage asynchronous dialog and to await the DATA, CREDIT, INTERUPT and MGCLOSED messages. If SEM is unspecified, the mailbox semaphore is used by default. If SEM is set to () or to zero, CAM selects a private POOL semaphore previously declared by H_CSEMPOOL.

	If SEM is neither () nor zero, it must specify a semaphore already declared as () or zero by a previous H_CENABLE, H_CINIT, or H_CACCEPT.
REQID	Request identifier (see subsection "REQID principles"). REQID is mandatory if the correspondent connects or is likely to connect to more than one session simultaneously. If REQID is not specified (single-session option only), it defaults to that specified at mailbox activation.
OPTION	Character string to be transmitted to the session initiator in the INQA area on execution of the H_CINQ primitive. This function is reserved for IOF connections or TDS connections.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
CONFLICT:	Attempt to mix a synchronous session with an asynchronous session on the same mailbox.
DAMAGED:	System error in buffer allocation.
DPREJCT:	Dialog error. Negotiation of dialog parameters has failed.
ENTRYOV:	No further POOL semaphores available.
FUNCNAV:	Attempt to select a non-existent semaphore.
PCREJCT:	Presentation error. Negotiation of presentation parameters has failed.

SYSOVLD: Insufficient system resources.

Comments

- The correspondent that receives a session opening request must execute H_CINQ to obtain session parameters before executing H_CACCEPT.
- All sessions initialized (H_CINIT) or accepted (H_CACCEPT) on the same mailbox must have the same connection mode, that is, they must all be synchronous or all asynchronous.

2.2.2 H_CATTN

Function

Sends an attention-type interrupt (a break) and asks the destinee to respond with a specific action.

An interrupt may be received in response.

Syntax

```
$H_CATTN

          SREF= i_fb15

          [,ITCODE = i_bit_16]

          ,ITA = o_structure;
```

Parameters

SREF — Session reference returned by CAM at session initiation.

ITCODE — Interrupt code specifying the action required of the destinee. Default value is "0000"X.

ITA — Interrupt reception zone with the following format:

```
DCL 1 ITA
      2 TYPE    CHAR(1),
      2 CODE    CHAR(2),
      2 SIZE    FIXED BIN(15),
      2 DATA    CHAR(140);
```

See subsection "Interrupts".

Normal return codes

DONE

INTERUPT:	Interrupt code received in the ITA zone. Requested action not done.

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.

Comments

The attention is sent regardless of the correspondent's current turn and credit status. Nor does H_CATTN modify this status.

2.2.3 H_CDISABLE

Function

Deactivates the mailbox. All sessions on the mailbox must first be closed.

Syntax

```
$H_CDISABLE
                imn;
```

Parameters

imn	Internal mailbox name (l_char16).

Normal return codes

DONE

ALREADY:	The mailbox is already disabled.

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
BUSY:	The mailbox cannot be disabled because one or more sessions are still open.

Comments

If the mailbox semaphore was specified at mailbox activation as SEM = () or zero, it is deleted from the private pool and cannot be used elsewhere. However if an existing semaphore was specified, the semaphore is not deleted.

2.2.4 H_CENABLE

Function

Activates the mailbox and selects its message paths.

Syntax

```
$H_CENABLE
          imn
          [,MBX = i_char_8]
          ,SEM = b_ptr
          ,REQID = i_bit_16
          [,MPLIST = (ipn[,ipn[,...]])]
          [,MAXS = i_fb15]
          [,NOCREDIT]
          [,APPLCHECK];
```

Parameters

imn — Internal mailbox name (l_identifier16). Normally this is a name declared by the primitive H_DCMBX. However, for compatibility with the previous version, the name is accepted even if it has not been declared by H_DCMBX.

MBX — External mailbox name. This is the name by which the mailbox is known to other correspondents, and so it must be unique within the site.

If MBX is not specified, it defaults to the name declared by the EXT_LMBX_NAME parameter of H_DCMBX.

SEM	Semaphore used to receive an OPENREQ or SHUTDOWN message. It may also be used to manage asynchronous dialog (see H_CACCEPT and H_CINIT). If SEM is set to () or to zero, CAM selects a semaphore from the private POOL declared by H_CSEMPOOL, or, if this does not exist, from the system POOL. For compatibility with the previous version, a private POOL is automatically created (with NBSEM=1 and NBMESS=MAXS*3) if the number of sessions is greater than or equal to 30. If SEM is neither () nor zero, it must specify a semaphore already declared as () or zero by a previous H_CENABLE or a semaphore specified by a previous H_CGETSEM.
REQID	Request identifier (see subsection "REQID principles").
NOCREDIT	If NOCREDIT is specified, the CREDIT semaphore message is not received either after session connection or after receipt of a LEVEL=3 or LEVEL=5 send.
APPLCHECK	If APPLCHECK is specified, access rights are checked at all session connections to this mailbox. Note that the CAM mailbox name must be cataloged, i.e. included in the application list (APPLIST) of the project requesting the connection. If APPLCHECK is not specified, only the identity of the requestor (if transmitted) is checked.
MPLIST	Set of possible message paths to be used when accepting a session. Each message path is identified by an internal path name (ipn) previously declared by H_DCMP.

	Only the message paths to be used for accepting sessions need be specified; the message path used when requesting a session is specified by H_CINIT. If the list of message paths is empty, all requests to open a session with the mailbox are rejected by CAM.
MPLIST (Cont'd)	The allocation attributes of a message path (INTERNAL, EXTERNAL, STATIC, AUTOMATIC) must allow H_WAIT internal access to the message path when the OPENREQ semaphore message arrives. The list of message paths must not include a message path already activated by a previous H_CENABLE. Once a message path has been selected by H_CENABLE, it cannot afterwards be modified.
MAXS	Maximum number of sessions that can be simultaneously connected to this mailbox. MAXS=1 is the default.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
BUSY:	Mailbox currently being activated or deactivated.
DUPNAME:	Mailbox already activated.
ENTRYOV:	No further POOL semaphores available.
SYSOVLD:	Insufficient system resources.

Comments

See subsection "Limits of Usage".

2.2.5 H_CGETSEM

Function

Allocates a semaphore from the private POOL of semaphores for use by the H_CENABLE, H_CINIT, H_CACCEPT and H_WAIT primitives.

Syntax

```
$H_CGETSEM

          SEM = o_ptr;
```

Parameters

SEM	Semaphore from the private POOL.

Return codes

ENTRYOV:	No further POOL semaphores available.
FUNCNAV:	Semaphore POOL non-existent.

Comments

- The private semaphore POOL must already exist (created by H_CSEMPOOL).
- The semaphore specified by H_CGETSEM is automatically returned to the system by CAM's exit routine at process group termination.

2.2.6 H_CINIT

Function

Requests a session with another mailbox. The submitter's mailbox must be active (that is, H_CINIT must be preceded by H_CENABLE).

The request is received when the destinee's mailbox is active and CAM is able to match the message path.

When the session is accepted by the destinee, the requestor must issue an H_CINQ. From this point, the session is established and dialog can begin.

Syntax

```
$H_CINIT
          imn
          ,MP = l_identifier16
          ,SREF = o_fb15
          [{,ASYNC [,SEM = b_ptr] [,REQID =i_bit_16] | ,SYNC }]
          [,DESTNODE = i_char_8, DESTMBX = i_char_8]
          [,SBMID = i_char_48]
          [,OPTION = i_char_23]
          [,STATION = i_char_8]
          [,NCONTROL]
          [,USERKNOWN];
```

Parameters

imn	Submitter's internal mailbox name; must be declared by H_DCMBX.
MP	Message path proposed by the submitter; must be declared by H_DCMP.
SREF	Session reference returned by CAM. Both correspondents specify this reference when executing the H_CINQ, H_CSEND, H_CRECEIVE, H_CTERM, H_CATTN, and H_CTELEG primitives in order to identify the relevant session.
ASYNC	Dialog between initiator and CAM is to be asynchronous. Mandatory in TWS mode.
SEM	Semaphore used in asynchronous dialog to await the OPENACK, DATA, CREDIT, INTERRUPT and MGCLOSED messages. If SEM is not specified, the mailbox semaphore (see H_CENABLE) is used instead. If SEM is set to () or to zero, CAM selects a private POOL semaphore previously declared by H_CSEMPOOL. If SEM is neither () nor zero, it must specify a semaphore already declared as () or zero by a previous H_CENABLE, H_CINIT, or H_CACCEPT.
REQID	Request identifier to be returned with each semaphore message, thus enabling the correspondent to identify the source of the message. REQID is mandatory if the correspondent connects or is likely to connect to more than one session simultaneously.

If REQID is not specified (single-session option only), it defaults to that specified at mailbox activation.

DESTNODE — Destinee node name. If DESTNODE is unspecified (left blank), the local node is assumed.

DESTMBX — Destinee mailbox name. If DESTNODE and DESTMBX are unspecified, they default to the values declared by DNODE and DMBX of the H_DCMP primitive.

SBMID — Submitter's user/billing/project/password identity declared as follows:

```
DCL 1 SBMID,
    2 BILLING     CHAR(12),
    2 USER        CHAR(12),
    2 PROJECT     CHAR(12),
    2 PASSWORD    CHAR(12);
```

BILLING and PROJECT do not need to be specified if they are the submitter's default project and billing. All empty fields must consist of blanks.

If all four fields are set to binary zero, the DSA security record is not transmitted in the connection letter.

If SBMID is unspecified, the user identity (and password) is set to that of the job submitter (default) or that specified by the job submission command.

OPTION — Character string to transmit to the target correspondent in the INQA zone, on execution of the H_CINQ primitive. This function is reserved for IOF or TDS sessions. For TDS connections, see the *TDS COBOL Programmer's Guide*.

STATION — Submitter's station name. This parameter is reserved for IOF connections.

NCONTROL	If NCONTROL is specified, there is no checking of the password given by SBMID.
USERKNOWN	Allows the user to be connected to IOF or TDS even if already connected to another session.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
CONFLICT:	Attempt to mix synchronous mode with asynchronous mode on the same mailbox.
DNSPEC:	Destinee mailbox not specified by either DESTMBX or DMBX.
ENTRYOV:	No further POOL semaphores available.
FUNCNAV:	Private POOL semaphore specified (SEM = () or zero) and semaphore POOL non-existent.
MBXSAT:	Maximum number of sessions (MAXS) already connected to the initiator mailbox.
SYSOVLD:	Insufficient system resources.

Abnormal return codes (synchronous mode only)

DAPPLSAT:	Destinee application saturated.
DMBXNOP:	Destinee mailbox disabled or in the process of being disabled.
DMBXSAT:	Destinee mailbox saturated; maximum number of sessions (MAXS) already connected to it.
DMBXUNKN:	Destinee mailbox unknown.

DNODNOP:	Destinee node unavailable.
DNODSAT:	Destinee node saturated.
DPREJCT:	Dialog error. Parameter negotiation has failed.
DUPUSER:	User is already logged on. Applicable to IOF and TDS connections only.
INITVIOL:	User does not have the necessary access rights to connect to the destinee mailbox.
NODUNKN:	Destinee node unknown.
PATHNAV:	Connection path cannot be opened (line not up, driver not ready, ...)
PCREJCT:	Presentation error. Parameter negotiation has failed.
SCTYVIOL:	Security violation. The specified USER, BILLING, PROJECT, and PASSWORD combination is not registered in the in the Site Catalog.
SESREJCT:	Session refused. No reason given.
TIMEOUT:	Connection time (CNCTIME) has expired while waiting for a response from the destinee mailbox. Applicable to TDS connections only (see *TDS COBOL Programmer's Guide*).
USERRJCT:	Session refused by the destinee mailbox. USERRJCT represents a class of return codes whose values fall within the range specified for the REASON parameter of the H_CREJECT primitive.

Comments

- The turn is defined by the message path returned by H_CINQ, fields TWAI and TWAA. Where the destinee mailbox is a terminal, the turn depends on the message path specified by the application (not the terminal manager). If the application does not select the turn option, the turn is given to the session initiator (see subsection "Session Negotiation")

- All sessions initialized (H_CINIT) or accepted (H_CACCEPT) on the same mailbox must have the same connection mode, that is, they must all be synchronous or all asynchronous.

2.2.7 H_CINQ

Function

Returns session parameters negotiated by CAM and also information about the partner correspondent.

An acceptor correspondent must issue H_CINQ before issuing H_CACCEPT; an initiator correspondent must issue H_CINQ before proceeding with the session dialog.

Syntax

```
$H_CINQ
          SREF = i_fb15
          ,INQA = b_structure;
```

Parameters

SREF	Session reference returned by CAM at session initiation.
INQA	Reception zone for the session parameters (see "Session Negotiation"). The zone must be declared by H_DCINQ.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".

Comments

- The number of DEVICE entries (i.e. ENTRYNB) must be initialized before H_CINQ is executed. Its value must be based on the NBENTRY parameter of the H_DCINQ primitive.
- H_CINQ must not be executed more than once by a session initiator.

2.2.8 H_CRECEIVE

Function

Receives data or interrupt codes.

Syntax

```
$H_CRECEIVE
          SREF = i_fb15
          ,WA = o_structure
          ,MAXLEN = i_fb15
          ,OUTLEN = o_fb15
          ,LEVEL = o_bit_8
          [,HDR = i_structure]
          ,ITA = o_structure;
```

Parameters

SREF	Session reference returned by CAM at session initiation.
WA	Data reception zone.
MAXLEN	Length of the WA zone.
OUTLEN	Length of the data received.
LEVEL	Data level: LEVEL=1 : data record LEVEL=3 : end-of-interaction (and get the turn) LEVEL=5 : end-of-session

	See subsections "Session Dialog" and "Session Termination".
HDR	Device header reception zone. See subsection "Device Header".
ITA	Interrupt reception zone. See subsection "Interrupts".

Normal return codes

DONE

IGNORE:	No information received.
INTERUPT:	Interrupt code received in the ITA zone.

Abnormal return codes

ARGERR:	Internal CAM error. See subsection "ARGERR Return Codes".
DAMAGED:	System error in buffer allocation.
RCVVIOL:	Attempt to receive data without first relinquishing the turn.
TRUNC:	Data reception zone (WA) is too short to contain the data received. The data is truncated and the excess is lost.

Comments

- OUTLEN et LEVEL are left empty if no data is received.
- With asynchronous dialog, H_CRECEIVE can be executed after a DATA or INTERUPT semaphore message only. If the interrupt is direct (no INTERUPT message first), the correspondent must wait for the DATA message before H_CRECEIVE can be executed.

2.2.9 H_CREJECT

Function

Rejects a request to open a session.

Syntax

```
$H_CREJECT
          SREF = i_fb15
          [,REASON = i_bit_16];
```

Parameters

SREF	Session reference returned by CAM at session initiation.
REASON	Reason for the rejection, to be transmitted to the session initiator. See the H_CINIT return codes where the dialog is synchronous, or OPCODE=OPENACK where the dialog is asynchronous. REASON must be in the range "000B"X to "0013"X. This corresponds to the USERRJCT class of return codes for H_CINIT. If REASON is unspecified or not in the above range, it is set to "0001"X. This corresponds to the SESREJCT of H_CINIT.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.

2.2.10 H_CSEMPOOL

Function

Creates a private pool of type 0 semaphores.

Syntax

```
$H_CSEMPOOL
          NBSEM = i_fb15
          NBMESS = i_fb15;
```

Parameters

NBSEM	Maximum number of semaphores that can be allocated in this pool.
NBMESS	Maximum number of semaphore messages available for the whole pool.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
FUNCNAV:	Semaphore POOL already allocated.

Comments

- One semaphore POOL only can be allocated per process group. Once the POOL is declared, the semaphores requested by H_CENABLE, H_CINIT and H_ACCEPT (with SEM = () or zero) are automatically allocated in this POOL. When the process group terminates, semaphore pool is returned to the system.
- For compatibility with the previous version, a semaphore POOL is created automatically for a mailbox connected to 30 or more sessions. The POOL is dimensioned with:
  - NBSEM=1
  - NBMESS= maximum number of sessions (MAXS) * 3
- Parameters NBSEM and MBMESS must observe the following constraints:
  - 40 + (16 * NBMESS) + (12 * NBSEM) < 64 k
  - NBMESS >= NBSEM * 3

2.2.11 H_CSEND

Function

Sends data to the partner correspondent. An interrupt might be received in response.

Syntax

```
$H_CSEND
          SREF = i_fb15
          ,WA = i_structure
          ,LENGTH = i_fb15
          ,LEVEL = i_bit_8
          [,HDR = i_structure]
          ,ITA = o_structure;
```

Parameters

SREF	Session reference returned by CAM at session initiation.
WA	Zone containing the data to be sent.
LENGTH	Length of the data to be sent.
LEVEL	Data level: LEVEL=1 : data record (and keep the turn). LEVEL=3 : end-of-interaction (and lose the turn) LEVEL=5 : end-of-session See subsections "Session Dialog" and "Session Termination".
HDR	Zone containing the DEVICE HEADER to be sent. See subsection "Device Header".

ITA	Interrupt reception zone. See subsection "Interrupts".

Normal return codes

DONE

INTERUPT:	Interrupt code received in the ITA zone. The data is not transmitted (but still available in WA).

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.
SNDVIOL:	Attempt to send data without having the turn.

Comments

In asynchronous dialog, the sender must obtain a CREDIT before each execution of H_CSEND.

2.2.12 H_CTELEG

Function

Sends telegram-type data to a correspondent. Used in exceptional cases where data cannot be sent in the normal way.

An interrupt may be received in response.

Syntax

```
$H_CTELEG
          SREF = i_fb15
          ,WA = i_structure
          ,LENGTH = i_fb15
          [,HDR = i_structure]
          ,ITA = o_structure;
```

Parameters

SREF	Session reference returned by CAM at session initiation.
WA	Zone containing the telegram to be sent.
LENGTH	Telegram length. The maximum length is 140 bytes, but for a given session it cannot exceed the RECORD size returned H_CINQ.
HDR	Zone containing the DEVICE HEADER to be sent. See the subsection "Device Header".
ITA	Interrupt reception zone. See subsection "Interrupts".

Normal return codes

DONE

INTERUPT:	Interrupt code received in the ITA zone. The telegram is not sent (but remains available in the WA zone).

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.

Comments

- The data is sent whether or not the correspondent has the turn and credit. H_CTELEG does not modify the turn or credit.
- This type of transmission is to be used only exceptionally. CAM does not allow repeated usage of H_CTELEG. Two consecutive telegrams cause a synchronous delay, because the first must be received before the second can be sent.
- If EXTENDED DATA ATTENTION protocol is negotiated (INQ_EXTDATN="1"B), the coding of the telegram's first byte is normalized by the protocol terminal manager.

2.2.13 H_CTERM

Function

Requests abnormal session termination.

Syntax

```
$H_CTERM
          SREF = i_fb15;
          [,ITCODE = i_bit_16];
```

Parameters

SREF	Session reference returned by CAM at session initiation.
ITCODE	Interrupt code to be transmitted to the destinee. Default value is "0000"X.

Normal return codes

DONE

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.

2.2.14 H_CTURN

Function

Requests the turn. This primitive is not allowed with TWS mode.

An interrupt can be received in response.

Requesting the turn sends an interrupt to the partner correspondent, who responds with a LEVEL=3 send to relinquish the turn.

Syntax

```
$H_CTURN

          SREF = i_fb15

          [,ITCODE = i_bit_16]

          ,ITA = o_structure;
```

Parameters

SREF	Session reference returned by CAM at session initiation.
ITCODE	Interrupt code, to be transmitted to the destinee. Default value is "0000"X.
ITA	Interrupt reception zone. See subsection "Interrupts".

Normal return codes

DONE:	In synchronous mode, this indicates that the turn is obtained and an H_CSEND can be executed. In asynchronous mode, this indicates that the turn is obtained and that an H_CSEND can be executed as soon as the CREDIT is received.
DATALIM:	Synchronous mode only. Turn obtained following a LEVEL=5 (end-of-session) send by partner.
ALREADY:	Submitter already has the turn.
INTERUPT:	Interrupt code received in the ITA zone. Request not sent.

Abnormal return codes

ARGERR:	CAM error. See subsection "ARGERR Return Codes".
SYSOVLD:	Insufficient system resources.

Comments

- Once an H_CTURN has been issued, neither correspondent receives any messages. The messages queued at both ends of the session are deleted by the system.
- The turn can be requested at any moment except when the requestor already has the turn or when session termination (normal or abnormal) is in progress.

2.2.15 H_DCCODE

Function

Declares a list of mnemonic indentifiers for OPCODE values and session rejection reasons returned in the H_WAIT semaphore message reception zone (see the MSG parameters of the H_WAIT primitive).

Syntax

```
$H_DCCODE [PREFIX = 1_identifier];
```

Parameters

PREFIX — Specifies a character string which is prefix of the %REPLACE name. Default value: empty string

Expansion

```
$H_DCCODE ;
                                        /* OPCODE VALUES          */
   %REPLACE OPENREQ       BY "01"X;
   %REPLACE OPENAK        BY "02"X;
   %REPLACE MGCLOSED      BY "03"X;
   %REPLACE DATA          BY "04"X;
   %REPLACE CREDIT        BY "05"X;
   %REPLACE INTERUPT      BY "06"X;
   %REPLACE SHUTDOWN      BY "09"X;

                                        /* OPEN REJECTION REASONS */
   %REPLACE SESREJCT      BY "0001"X;
   %REPLACE DNODNOP       BY "0002"X;
   %REPLACE DNODSAT       BY "0003"X;
   %REPLACE DMBXUNKN      BY "0004"X;
   %REPLACE DMBXNOP       BY "0005"X;
   %REPLACE DMBXSAT       BY "0006"X;
   %REPLACE DAPPLSAT      BY "0007"X;
   %REPLACE DPREJCT       BY "0009"X;
   %REPLACE PCREJCT       BY "000A"X;
   %REPLACE TIMEOUT       BY "0015"X;
   %REPLACE INITVOL       BY "0017"X;
   %REPLACE SCTYVIOL      BY "0018"X;
   %REPLACE DNODUNKW      BY "0040"X;
   %REPLACE PATHNAV       BY "0041"X;
   %REPLACE DUPUSER       BY "0042"X;
   %REPLACE CLSRJCT       BY "0044"X;
```

```
%REPLACE DAPPGT        BY "0045"X;
```

2.2.16 H_DCINQ

Function

Declares the zone for receiving the connection parameters generated at session initiation. It contains the following information:

- the identity of the session initiator
- negotiated parameter values (see subsection "Session Negotiation")
- the non-negotiable parameters proposed by the partner correspondent
- node and mailbox names of the acceptor

This information is retrieved by each of the correspondents when they execute H_CINQ.

Syntax

```
$H_DCINQ
          [,ATTRIB = {l_char | NLVL1}]
          [,NBENTRY = l_digit_1]
          [,PREFIX  = l_identifier];
```

Parameters

ATTRIB	GPL attribute for the structure.
NVL1	If NLVL1 is specified, the characters "DCL" and the end-of-structure semicolon are not generated. The structure starts at level 4.
NBENTRY	Maximum number of DEVICE-type entries to be reserved: 1<=NBENTRY<=3
PREFIX	Field name prefix.

Expansion

```
$H_DCINQ  NBENTRY = 3, PREFIX = IN1_, ATTRIB = STATIC;

/***********************************
            IN1_INQ
************************************/
DCL 1       IN1_INQ  STATIC,
    4  IN1_SUBMITT                    ,/*                          */
    5  *              CHAR( 2)        ,/*                          */
    5  IN1_IDENTS                     ,/*                          */
    6  IN1_BILLING    CHAR(12)        ,/*                          */
    6  IN1_USER       CHAR(12)        ,/*                          */
    6  IN1_PROJECT    CHAR(12)        ,/*                          */
    5  *              CHAR(39)        ,/*                          */
    4  IN1_LOGOPTN    CHAR(23)        ,/*                          */
    4  *              CHAR(33)        ,/*                          */
    4  IN1_MPPART                     ,/*                          */
    5  *              CHAR( 9)        ,/*                          */
    5  IN1_MGPART                     ,/*MSG GROUP PART            */
    6  IN1_MODE                       ,/*MSG GROUP MODE            */
    7  IN1_TWAI       BIT( 1)         ,/*TWA INITIATOR TURN        */
    7  IN1_TWAA       BIT( 1)         ,/*TWA ACCEPTOR TURN         */
    7  *              BIT( 4)         ,/*                          */
    7  IN1_TWS        BIT( 1)         ,/*TWO WAY SIMULTANEOUS      */
    7  *              BIT( 1)         ,/*                          */
    6  IN1_INTEGTY                    ,/*                          */
    7  *              BIT( 5)         ,/*                          */
    7  IN1_XCP1       BIT( 1)         ,/*XCP1 SESSION              */
    7  *              BIT( 1)         ,/*                          */
    7  IN1_ACMBXEX    BIT( 1)         ,/*ACPT MAILBOX EXTENSION    */
    6  *              BIT(16)         ,/*                          */
    6  IN1_IRECSZ     FIXED BIN(15)   ,/*MAX INPUT SIZE            */
    6  IN1_ORECSZ     FIXED BIN(15)   ,/*MAX OUTPUT SIZE           */
    6  IN1_ITRAFFIC   BIT( 8)         ,/*TRAFFIC CLASS             */
    6  IN1_OTRAFFIC   BIT( 8)         ,/*TRAFFIC CLASS             */
    6  IN1_CNCTIME    BIT( 8)         ,/*CONNECTION QUEUEING TIME*/
    5  IN1_ADDPART                    ,/*ADDRESSING PART           */
    6  IN1_LOCAL                      ,/*                          */
    7  *              CHAR( 4)        ,/*                          */
    7  IN1_LMBXEXT    CHAR( 4)        ,/*LOCAL MAILBOX EXTENSION*/
    7  *              CHAR( 8)        ,/*                          */
    6  IN1_DESTAD                     ,/*                          */
    7  IN1_DNODE      CHAR( 4)        ,/*DESTINATION NODE NAME     */
    7  IN1_DMBXEXT    CHAR( 4)        ,/*DEST MAILBOX EXTENSION    */
    7  IN1_DMBX       CHAR( 8)        ,/*DESTINATION MBOX NAME     */
    7  *              BIT( 8)         ,/*                          */
    4  IN1_APPL                       ,/*APPLICATION PART          */
    5  IN1_LVL        BIT( 4)         ,/*CAM VERSION               */
    5  *              BIT( 4)         ,/*                          */
    5  IN1_OPTION                     ,/*                          */
    6  *              BIT( 1)         ,/*                          */
    6  IN1_DVCHDR     BIT( 2)         ,/*DEVICE PROCEDURE HEADER*/
    6  *              BIT( 3)         ,/*                          */
    6  IN1_EXTDATN    BIT( 1)         ,/*EXTENDED DATA ATTN USED*/
    6  *              BIT( 1)         ,/*                          */
```

```
6  IN1_MODEL      FIXED BIN(15)  ,/*VISIBILITY              */
6  IN1_CODE                      ,/*PRESENTATION CODE       */
7  *              BIT( 6)        ,/*                        */
7  IN1_EBCDIC     BIT( 1)        ,/*                        */
7  IN1_ASCII      BIT( 1)        ,/*                        */
7  IN1_BINARY     BIT( 1)        ,/*                        */
7  *              BIT( 7)        ,/*                        */
6  IN1_SYMBOLS                   ,/*CHARACTER SET           */
7  *              BIT( 3)        ,/*                        */
7  IN1_SET96      BIT( 1)        ,/*                        */
7  IN1_SET64      BIT( 1)        ,/*                        */
7  *              BIT( 3)        ,/*                        */
5  IN1_ENTRYNB    BIT( 8)        ,/*DEVICE ENTRY NUMBER     */
4  IN1_DVCD                      ,/*DVC DESC PART           */
5  IN1_DVENTRY    (3)            ,/*                        */
6  IN1_PAGEL      BIT( 8)        ,/*PAGE LENGTH             */
6  IN1_LINEL      BIT( 8)        ,/*LINE LENGTH             */
6  IN1_DVCAP                     ,/*DEVICE CAPABILITY       */
7  IN1_INPUT      BIT( 1)        ,/*                        */
7  IN1_OUTPUT     BIT( 1)        ,/*                        */
7  *              BIT( 1)        ,/*                        */
7  IN1_SLAVE      BIT( 1)        ,/*                        */
7  *              BIT( 4)        ,/*                        */
6  IN1_DVTYPE                    ,/*DEVICE TYPE             */
7  IN1_DISPLAY    BIT( 1)        ,/*                        */
7  IN1_KEYBRD     BIT( 1)        ,/*                        */
7  IN1_PRINTER    BIT( 1)        ,/*                        */
7  IN1_DISKET     BIT( 1)        ,/*                        */
7  IN1_CASSET     BIT( 1)        ,/*                        */
7  *              BIT( 3)        ,/*                        */
6  IN1_DVMODEL    FIXED BIN(15)  ,/*DEVICE MODEL            */
6  IN1_DVFEATR                   ,/*DEVICE FEATURES         */
7  *              BIT( 2)        ,/*                        */
7  IN1_ROLLUP     BIT( 1)        ,/*AUTO ROLLING UP         */
7  IN1_WRAPAR     BIT( 1)        ,/*AUTO WRAP AROUND        */
7  *              BIT( 2)        ,/*                        */
7  IN1_LINFEED    BIT( 1)        ,/*AUTO LINE FEED          */
7  IN1_UNFOLD     BIT( 1)        ,/*AUTO LINE FOLD          */
7  *              BIT( 6)        ,/*                        */
7  IN1_HTAB       BIT( 1)        ,/*HORIZONTAL TABULATION   */
7  IN1_VTAB       BIT( 1)        ,/*VERTICAL TABULATION     */
6  *              PTR ;           /*ADR OF NEXT ENTRY IN MPD*/
```

Field Description

BILLING	Billing identity associated with the session initiator (see H_CINIT).
USER	User identity associated with the session initiator (see H_CINIT).
PROJECT	Project identity associated with the session initiator (see H_CINIT).
LOGOPTN	Character string specified by the session initiator or the session acceptor. This feature is reserved for IOF or TDS connections. For TDS connections, see the *TDS COBOL Programmer's Guide*).
MODE	Data exchange mode for the session. This option is negotiated under Rule 1 (see subsection "Session Negotiation"). If TWAI=1, session control alternates between the two correspondents with the initiator getting the first turn. If TWAA=1, session control alternates between the two correspondents with the acceptor getting the first turn. If TWS=1, the exchange is in both directions simultaneously. There is no "turn" concept.
XCP1	If XCP1=1, the session is an XCP1 session with reserved usage. If XCP1=0, the session is a normal session.
ACMBXEX	If ACMBXEX=1, mailbox extensions are used. See parameters LMNXEXT and DMBXEXT.
IRECSZ	Maximum record size that can be received (session layer parameter given for information). This option is negotiated under Rule 2 (see subsection "Session Negotiation").

ORECSZ	Maximum record size that can be sent (session layer parameter given for information). This option is negotiated under Rule 2 (see subsection "Session Negotiation").
CNCTIME	Connection wait time. There are two values possible: 0 = the initiator does not wait for the response. 31 = the initiator waits indefinitely for the response.
LMBXEXT	Initiator's mailbox extension name.
DNODE	Destinee's node name.
DMBXEXT	Destinee's mailbox extension name.
DMBX	Destinee's mailbox name.
LVL	CAM version number (internal parameter).
DVHDR	If DVHDR=00, the DEVICE HEADER option is not used. If DVHDR=01, the DEVICE HEADER option must be specified by the HDR parameter of the H_CSEND, H_CRECEIVE and H_CTELEG primitives. This option is negotiated under Rule 3 (see subsection "Session Negotiation")
EXTDATN	If EXTDATN=1, the EXTENDED DATA ATTENTION protocol is used (see H_CTELEG).
MODEL	See H_DCMODEL for list of possible values.
CODE	Presentation code: EBCDIC, ASCII, or binary. This option is negotiated under Rule 1 (see subsection "Session Negotiation")
SYMBOLS	Character set: 96 characters or 64 characters. This option is negotiated under Rule 1 (see subsection "Session Negotiation")

ENTRYNB	Number of initialized device entries. Each entry gives the parameters specific to the terminal that the correspondent can access. Meaningful only if MODEL is of type terminal.
PAGEL	Number of lines per page for the device concerned.
LINEL	Number of characters per line for the device concerned.
DVCAP	Device I/O capability (input, output, or input/output). The capability is defined by the corresponding bit being set to 1.
SLAVE	If SLAVE=1, the terminal is a SLAVE correspondent. This option is applicable to TDS terminals only (see the *TDS COBOL Programmer's Guide*).
DVTYPE	Device type (screen, keyboard, printer, diskette, or cartridge).
DVMODEL	Device model. See H_DCMODEL for list of possible values.
DVFEATR	If set to 1, the device features (automatic roll-up, wrap around, line feed, line fold, horizontal tabulation and vertical tabulation) are available.

2.2.17 H_DCMBX

Function

Declares the static structure for the mailbox. The structure links an internal mailbox identifier (NAME) with the external mailbox name (EXT_LMBX_NAME).

Syntax

```
$H_DCMBX

          NAME = l_identifier16

          [{,ATTRIB = l_char | ,NLVL1}]

          [,PREFIX = l_identifier4];
```

Parameters

NAME	Internal mailbox identifier
ATTRIB	GPL attribute for the structure.
NLVL1	If NLVL1 is specified, the characters "DCL" and the end-of-structure semicolon are not generated. The structure starts at level 3.
NBENTRY	Maximum number of DEVICE-type entries to be reserved: 1<=NBENTRY<=3
PREFIX	Field name prefix.

Expansion

```
$H_DCMBX   NAME=INT_LMBX_NAME, PREFIX=MB_, ATTRIB=STATIC;

/************************
        H_DCMBX
 ************************/
DCL 1 INT_LMBX_NAME STATIC,
    4 MB_EXT_LMBX_NAME CHAR(8),       /* EXTERNAL MAILBOX NAME  */
    4 *                FIXED BIN(15), /* RESERVED               */
    4 LVL              BIT(4);        /* RFU                    */
```

Field Description

EXT_LMBX_NAME — External mailbox name. This is the name by which the mailbox is known to its correspondents, and so must be unique within the site.

Comments

If both ATTRIB and NLVL1 are unspecified, H_DCMBX generates the structure with the EXTERNAL attribute. This maintains compatibility with the previous version and also conserves the default path from one compile unit to another.

2.2.18 H_DCMODEL

Function

Declares a list of mnemonic identifiers for a variety of terminal, application and processor models. These mnemonic identifiers can be specified for the MODEL and DVMODEL fields of the H_DCMP and H_DCINQ primitives.

Syntax

```
$H_DCMODEL;
```

Parameters

None.

Expansion

```
%REPLACE V_APPL          BY "1000"X;
%REPLACE HL61            BY "1001"X;
%REPLACE HL62            BY "1002"X;
%REPLACE HL64            BY "1003"X;
%REPLACE HL66            BY "1004"X;
%REPLACE IRIS80          BY "1005"X;
%REPLACE IRIS60          BY "1006"X;
%REPLACE TTY33           BY "2001"X;
%REPLACE TTY35           BY "2002"X;
%REPLACE TTY37           BY "2003"X;
%REPLACE TTY38           BY "2004"X;
%REPLACE TN300           BY "2005"X;
%REPLACE TE318           BY "2006"X;
%REPLACE DTU7170         BY "2007"X;
%REPLACE DTU7171         BY "2008"X;
%REPLACE DTU7172         BY "2009"X;
%REPLACE TTU8124         BY "200A"X;
%REPLACE TTU8126         BY "200B"X;
%REPLACE TTU8128         BY "200C"X;
%REPLACE VIP7100         BY "200D"X;
%REPLACE VIP7200         BY "200E"X;
%REPLACE TRC2811         BY "200F"X;
%REPLACE IBM2741         BY "2010"X;
%REPLACE TN1200          BY "2011"X;
%REPLACE TCV260          BY "2012"X;
```

```
%REPLACE TC349          BY "2013"X;
%REPLACE TC380          BY "2014"X;
%REPLACE AJ832          BY "2015"X;
%REPLACE VTE2820        BY "2016"X;
%REPLACE VIP7801        BY "2017"X;
%REPLACE IRISCOPE200    BY "2018"X;
%REPLACE DKU7001        BY "2019"X;
%REPLACE DKU7002        BY "201A"X;
%REPLACE DKU7102        BY "201B"X;
%REPLACE AJ833          BY "201D"X;
%REPLACE TI700          BY "201E"X;
%REPLACE AJ510          BY "201F"X;
%REPLACE TTX80          BY "2020"X;
%REPLACE TELEX          BY "2021"X
%REPLACE MINITEL1B      BY "202A"X;
%REPLACE VIP765         BY "2030"X;
%REPLACE VIP775         BY "2031"X;
%REPLACE VIP785         BY "2032"X;
%REPLACE VIP7700        BY "2033"X;
%REPLACE MTS7500        BY "2034"X;
%REPLACE BTT7340        BY "2035"X;
%REPLACE KDS7255        BY "2036"X;
%REPLACE KDS7265        BY "2037"X;
%REPLACE KDS7275        BY "2038"X;
%REPLACE TTU8221        BY "2039"X;
%REPLACE VIP7760        BY "203A"X;
%REPLACE TTS7800        BY "203B"X;
%REPLACE VIP7001        BY "203C"X;
%REPLACE DKU7005        BY "203D"X;
%REPLACE VIP7804        BY "203E"X;
%REPLACE DKU7008        BY "203F"X;
%REPLACE DKU7007        BY "2040"X;
%REPLACE DKU7105        BY "2041"X;
%REPLACE DKU7107        BY "2042"X;
%REPLACE TTX35S         BY "2043"X;
%REPLACE DKU7006        BY "2044"X;
%REPLACE DKU7211        BY "2045"X;
%REPLACE VIP7813        BY "2046"X;
%REPLACE VIP7814        BY "2047"X;
%REPLACE VIP7305        BY "2048"X;
%REPLACE HDS7           BY "204D"X;
%REPLACE DKU7211D       BY "204E"X;
%REPLACE VIP8800        BY "204F"X;
%REPLACE IRISCOPE300    BY "2050"X;
%REPLACE STS2840        BY "2051"X;
%REPLACE RBCOMP         BY "2052"X;
%REPLACE RBUNCOMP       BY "2053"X;
%REPLACE TL15MSV2       BY "2054"X;
%REPLACE IBM3741        BY "2060"X;
%REPLACE IBM3270        BY "2061"X;
%REPLACE IBM3278        BY "2064"X;
%REPLACE IBM3279        BY "2069"X;
%REPLACE IBM3287        BY "2067"X;
%REPLACE PC7800         BY "2029"X;
%REPLACE MT281          BY "205C"X;
%REPLACE TW52255        BY "2090"X;
```

2.2.19 H_DCMP

Function

Declares a message path containing all the options necessary for session dialog.

All unused parameters (documented and undocumented) must be set to binary zero.

Syntax

```
$H_DCMP
          [,ATTRIB = {l_char | NLVL1}]
          [,NBENTRY= l_digit_1]
          [,PREFIX= l_identifier];
```

Parameters

ATTRIB	GPL attribute for the structure.
NLVL1	If NLVL1 is specified, the characters "DCL" and the end-of-structure semicolon are not generated. The structure starts at level 4.
NBENTRY	Maximum number of DEVICE-type entries to be reserved: 1<=NBENTRY<=3
PREFIX	Field name prefix.

Expansion

```
$H_DCMP  NBENTRY = 3, PREFIX = MP1_, ATTRIB = STATIC;

/**********************************
               MP1_MP
 **********************************/
DCL 1 MP1_MP STATIC,
    4 MP1_MPPART                ,/*                          */
    5 *           CHAR( 9)      ,/*                          */
    5 MP1_MGPART                ,/*MSG GROUP PART            */
    6 MP1_MODE                  ,/*MSG GROUP MODE            */
    7 MP1_TWAI    BIT( 1)       ,/*T-W ALT INITIATOR TURN    */
    7 MP1_TWAA    BIT( 1)       ,/*T-W ALT ACCEPTOR TURN     */
    7 *           BIT( 4)       ,/*                          */
    7 MP1_TWS     BIT( 1)       ,/*TWO WAY SIMULTANEOUS      */
    7 *           BIT( 1)       ,/*                          */
    6 MP1_INTEGTY               ,/*                          */
    7 *           BIT( 5)       ,/*                          */
    7 MP1_XCP1    BIT( 1)       ,/*XCP1 SESSION              */
    7 *           BIT( 1)       ,/*                          */
    7 MP1_ACMBXEX BIT( 1)       ,/*ACCEPT MAILBOX EXTENSION*/
    6 *           BIT(16)       ,/*                          */
    6 MP1_IRECSZ  FIXED BIN(15),/*MAX INPUT RECORD SIZE     */
    6 MP1_ORECSZ  FIXED BIN(15),/*MAX OUTPUT RECORD SIZE    */
    6 MP1_ITRAFFC BIT( 8)       ,/*TRAFFIC CLASS             */
    6 MP1_OTRAFFC BIT( 8)       ,/*TRAFFIC CLASS             */
    6 MP1_CNCTIME BIT( 8)       ,/*CONNECTION QUEUEING TIME*/
    5 MP1_ADDPART               ,/*ADDRESSING PART           */
    6 MP1_LOCAL                 ,/*                          */
    7 *           CHAR( 4)      ,/*                          */
    7 MP1_LMBXEXT CHAR( 4)      ,/*LOCAL MAILBOX EXTENSION */
    7 *           CHAR( 8)      ,/*                          */
    6 MP1_DESTAD                ,/*                          */
    7 MP1_DNODE   CHAR( 4)      ,/*DESTINATION NODE NAME     */
    7 MP1_DMBXEXT CHAR( 4)      ,/*DEST MAILBOX EXTENSION    */
    7 MP1_DMBX    CHAR( 8)      ,/*DESTINATION MAILBX NAME  */
    7 *           BIT( 8)       ,/*                          */
    4 MP1_APPL                  ,/*APPLICATION PART          */
    5 MP1_LVL     BIT( 4)       ,/*CAM VERSION               */
    5 *           BIT( 4)       ,/*                          */
    5 MP1_OPTION                ,/*                          */
    6 *           BIT( 1)       ,/*                          */
    6 MP1_DVCHDR  BIT( 2)       ,/*DEVICE PROCEDURE HEADER  */
    6 *           BIT( 3)       ,/*                          */
    6 MP1_EXTDATN BIT( 1)       ,/*EXTENDED DATA ATTN USED  */
    6 *           BIT( 1)       ,/*                          */
    6 MP1_MODEL   FIXED BIN(15),/*VISIBILITY                */
    6 MP1_CODE                  ,/*PRESENTATION CODE         */
    7 *           BIT( 6)       ,/*                          */
    7 MP1_EBCDIC  BIT( 1)       ,/*                          */
    7 MP1_ASCII   BIT( 1)       ,/*                          */
    7 MP1_BINARY  BIT( 1)       ,/*                          */
    7 *           BIT( 7)       ,/*                          */
    6 MP1_SYMBOLS               ,/*CHARACTER SET             */
    7 *           BIT( 3)       ,/*                          */
```

```
7 MP1_SET96    BIT( 1)       ,/*                          */
7 MP1_SET64    BIT( 1)       ,/*                          */
7 *            BIT( 3)       ,/*                          */
5 MP1_ENTRYNB  BIT( 8)       ,/*DEVICE ENTRY NUMBER       */
4 MP1_DVCD                   ,/*DVC DESC PART             */
5 MP1_DVENTRY  (3),          ,/*                          */
6 MP1_PAGEL    BIT( 8)       ,/*PAGE LENGTH               */
6 MP1_LINEL    BIT( 8)       ,/*LINE LENGTH               */
6 MP1_DVCAP                  ,/*DEVICE CAPABILITY         */
7 MP1_INPUT    BIT( 1)       ,/*                          */
7 MP1_OUTPUT   BIT( 1)       ,/*                          */
7 *            BIT( 1)       ,/*                          */
7 MP1_SLAVE    BIT( 1)       ,/*                          */
7 *            BIT( 4)       ,/*                          */
6 MP1_DVTYPE                 ,/*DEVICE TYPE               */
7 MP1_DISPLAY  BIT( 1)       ,/*                          */
7 MP1_KEYBRD   BIT( 1)       ,/*                          */
7 MP1_PRINTER  BIT( 1)       ,/*                          */
7 MP1_DISKET   BIT( 1)       ,/*                          */
7 MP1_CASSET   BIT( 1)       ,/*                          */
7 *            BIT( 3)       ,/*                          */
6 MP1_DVMODEL  FIXED BIN(15),/*DEVICE MODEL               */
6 MP1_DVFEATR                ,/*DEVICE FEATURES           */
7 *            BIT( 2)       ,/*                          */
7 MP1_ROLLUP   BIT( 1)       ,/*AUTO. ROLLING UP          */
7 MP1_WRAPAR   BIT( 1)       ,/*AUTO. WRAP AROUND         */
7 *            BIT( 2)       ,/*                          */
7 MP1_LINFEED  BIT( 1)       ,/*AUTO. LINE FEED           */
7 MP1_UNFOLD   BIT( 1)       ,/*AUTO. LINE FOLD           */
7 *            BIT( 6)       ,/*                          */
7 MP1_HTAB     BIT( 1)       ,/*HORIZONTAL TABULATION     */
7 MP1_VTAB     BIT( 1)       ,/*VERTICAL TABULATION       */
6 *            PTR ;          /*ADR OF NEXT ENTRY IN MPD*/
```

Field Descriptions

MODE	Data exchange mode for the session. This option is negotiated under Rule 1 (see subsection "Session Negotiation") If TWAI=1, session control alternates between the two correspondents with the initiator getting the first turn. If TWAA=1, session control alternates between the two correspondents with the acceptor getting the first turn. If TWS=1, the exchange is in both directions simultaneously. Records are transmitted with LEVEL=1, 3, or 5 in the usual way but there is no waiting for a CREDIT (this is obtained systematically) and there is no "turn" concept.
XCP1	If XCP1=1, the session is an XCP1 session with reserved usage. If XCP1=0, the session is a normal session.
ACMBXEX	If ACMBXEX=1, mailbox extensions are used. Parameters LMBXEXT and DMBXEXT must be specified.
IRECSZ	Maximum expected size for messages received. This option is negotiated under Rule 2 (see subsection "Session Negotiation")
ORECSZ	Maximum expected size for messages sent. Same negotiation rule as for IRECSZ.
OTRAFFC/ITRAFFC	Expected level (or window) of message emission/reception. Only the value 1 is supported: 1 = one message only (window of 1)
CNCTIME	Connection wait time. There are two values possible: 0 = the initiator does not wait for the response.

	31 = the initiator waits indefinitely for the response. This parameter is reserved for TDS connections (see the *TDS COBOL Programmer's Guide*).
LMBXEXT	Extension name for the local mailbox. Used if ACMBXEX=1 and the parameters DESTNODE and DESTMBX of the primitive H_CINIT are unspecified.
DNODE	Destinee's node name. If set to 8 blanks, the local node is assumed.
DMBXEXT	Extension name for the destinee mailbox. Used if ACMBXEX=1 and the parameters DESTNODE and DESTMBX of the primitive H_CINIT are unspecified.
DMBX	Destinee's mailbox name.
LVL	CAM version number (internal parameter). Must be initialized at "0000"B.
DVCHDR	DEVICE HEADER option for use in dialog with a terminal (see subsection "Device Header"). There are three values possible: 11 = DEVICE HEADER mandatory 01 = DEVICE HEADER supported 00 = DEVICE HEADER not supported This option is negotiated under Rule 3 (see subsection "Session Negotiation")
EXTDATN	If EXTDATN=1, the EXTENDED DATA ATTENTION protocol is used (see H_CTELEG). This option is negotiated under Rule 1b (see subsection "Session Negotiation")
MODEL	Type of destinee (terminal, application, processor). See H_DCMODEL for list of possible values.

CODE	Presentation code: EBCDIC, ASCII, or binary. This option is negotiated under Rule 1 (see subsection "Session Negotiation")
SYMBOLS	Character set: 96 characters or 64 characters. This option is negotiated under Rule 1 (see subsection "Session Negotiation") if CODE=EBCDIC.
ENTRYNB	Number of initialized device entries. Each entry gives the parameters specific to the terminal that the correspondent can access. Meaningful only if MODEL is of type terminal.
PAGEL	Number of lines per page for the device concerned.
LINEL	Number of characters per line for the device concerned.
DVCAP	Device I/O capability (input, output, or input/output). The capability is defined by setting the corresponding bit to 1.
SLAVE	If SLAVE=1, the terminal is a SLAVE correspondent. This option is applicable to TDS terminals only (see the *TDS COBOL Programmer's Guide*).
DVTYPE	Device type (screen, keyboard, printer, diskette, or cartridge).
DVMODEL	Device model. See H_DCMODEL for list of possible values.
DVFEATR	Device features (automatic roll-up, wrap around, line feed, line fold, horizontal tabulation and vertical tabulation).

NOTE:

The internal name of the message path (ipn) is "<prefix>MP".

2.2.20 H_WAIT

Function

Used to wait for a semaphore event such as:

- a response to a session opening request (H_CINIT followed by H_WAIT), when dialog is asynchronous
- a session opening request (H_CENABLE followed by H_WAIT)
- a data transaction (H_WAIT followed by H_CSEND or H_CRECEIVE), when dialog is asynchronous
- an external interrupt (timer, OMH, ...)

Syntax

```
$H_WAIT
          SEM = i_ptr
          ,REQID = o_bit_16
          {,MSG48 = o_bit_48 | ,MSG = o_bit_32}
          [,TEST];
```

Parameters

SEM	Semaphore address (see H_CENABLE, H_CINIT and H_CACCEPT).
REQID	Request identifier specified for H_CENABLE, H_CINIT, or H_CACCEPT. If awaiting an OPENREQ or SHUTDOWN message, the REQID is that specified for H_CENABLE. If awaiting a dialog-related message, the REQID is that specified at H_CINIT and H_CACCEPT. Where REQID is not specified at session opening (single session only option), REQID defaults to that specified at H_CENABLE.

The OPENREQ and SHUTDOWN messages can be received on the mailbox semaphore only. The other messages are received on the session semaphore, or, if this is unspecified, at the mailbox semaphore.

MSG/MSG48 — Reception zone for the semaphore message, with the following format:

```
DCL 1 SEM_MSG,
    2 OPCODE      BIT(8),
    2 BYTE1       BIT(8),
    2 BYTE2_3,
      3 BYTE2     BIT(8),
      3 BYTES3    BIT(16),
    2 BYTES5_6    BIT(16);
```

TEST — If TEST is specified, there is no waiting for the semaphore event if it does not arrive.

Normal return codes

DONE

EMPTY: — TEST specified and no message received on the semaphore. Processing continues without waiting.

Abnormal return codes

ARGERR: — CAM error. See subsection "ARGERR Return Codes".

OPCODE

01 (OPENREQ): — Request to open a session. The receiver of this request must execute H_CINQ to negotiate the message path and then accept or refuse the connection (H_CACCEPT or H_CREJECT). BYTE1 and BYTE2 contain the SREF returned by CAM.

02 (OPENACK): Response to the request to open a session. If BYTE1<>0, the session is accepted. If BYTE1=1, the session is refused with the reason contained in BYTE2 and BYTE3. Reasons "0001" to "000A" are returned by CAM. Reasons "000B" to "0013" are returned by the refuser correspondent. Reasons "0015" and above are returned by the system.

The following values correspond to the GPL expression: STRING (BYTES2_3). Note that BYTES2_3 is a member of the SEM-MSG structure.

0001 (SESREJCT):	Session rejected with no reason specified.
0002 (DNODNOP):	Requested node unavailable
0003 (DNODSAT):	Requested node saturated
0004 (DMBXUNKN):	Requested mailbox not known
0005 (DMBXNOP):	Requested mailbox disabled or in the process of being disabled.
0006 (DMBXSAT):	Requested mailbox saturated (MAXS sessions already connected)
0007 (DAPPLSAT):	Requested application saturated.
0009 (DPREJCT):	Failure to negotiate dialog parameters.
000A (PCREJCT):	Failure to negotiate presentation parameters.
000B to 0013 (USERRJCT):	Reason specified by the refuser correspondent (see H_CREJECT).

0015 (TIMEOUT):	Response not received within the time specified by CNCTIME. Applicable to TDS connections only (see the *TDS COBOL Programmer's Guide*).
0017 (INITVIOL):	Requestor mailbox does not have the necessary access rights to open a session with the requested mailbox.
0018 (SCTYVIOL):	Security violation. The USER, PROJECT, BILLING, PASSWORD combination specified by the requestor does not exist in the Site Catalog.
0040 (DNODUNKW):	Requested node unknown.
0041 (PATHNAV):	Logical connection not possible (line not ready, device not ready, ...)
0042 (DUPUSER):	Requestor already "logged on". Applicable to IOF and TDS connections only.
0044 (CLSRJCT):	Connection refused because of an interrupt by H_CTERM or because control is passed to an EXIT ROUTINE (following an abort, for example).
0045 (DAPPPGT):	Requested application is at process group termination.

03	(MGCLOSED):	Session terminated abnormally (asynchronous mode only).
04	(DATA):	Data is available. An H_CRECEIVE must follow.
05	(CREDIT):	Buffer space is available; an H_CSEND can be executed. In asynchronous dialog, BYTE1 contains the response level (3 or 5) when CREDIT follows a request for the turn.
06	(INTERUPT):	An ITA interrupt has occurred. An H_CRECEIVE must follow.
09	(SHUTDOWN):	Session terminated by the telecommunications supervisor (command TTSVR TNS/FNPS/FEPS). An H_CDISABLE must follow.

A list of mnemonic identifiers for OPCODE values and session rejection reasons may be declared using the H_DCCODE primitive.

NOTE:

In the TDS environment, H_WAIT can block a simultaneity level. This can be avoided by specifying the TEST parameter. If as a result the return code is EMPTY, then come out of the TPR with a value for WAIT_TIME, and then re-execute H_WAIT after this period of time.

2.3 Dynamic Operator Command Primitives

2.3.1 H_CRCOM

Function

Creates a new operator command.

Syntax

```
$H_CRCOM
          name
          ,SEM = i_ptr
          ,REQID = i_bit_16
          [,PRIORITY = i_fb15]
          [,SYSTEM] ;
```

Parameters

name	Name of the command to be created (i_char4)
SEM	Pointer on the semaphore that will be notified each time an operator types the created command.
REQID	Request identifier (see subsection "REQID principles").
PRIORITY	Enqueueing priority of the messages with which the specified semaphore will be notified. If not specified, the default value is 15.
SYSTEM	If specified, the created command is a system command. Any console or terminal under IOF can

access it. If not specified, the command is accessible only from the submitter's console.

Normal return codes

DONE

Abnormal return codes

OPQOVLD:	Request rejected because of temporary overload of system queue.
CDERR:	System reserved command name or command already created for that console (if SYSTEM is not specified).
DUPNAME:	Command name already created.

Comments

To this command is attached a semaphore. The process, which created the command, waits on this semaphore for the notification of the event. The semaphore message contains the event type and the request identifier. The waiting mechanism is provided by the H_WAIT primitive or by the GPL built-in functions SEPM, SEPTM, ...

The semaphore message is a 16-byte area with the following structure:

```
DCL 1  SEM_MSG,
    2  RFU1     CHAR(10),
    2  REQID    FIXED BIN (15),
    2  RFU2     CHAR(4);
```

If after the P-operation the retrieved REQID is equal to the request identifier defined at the creation of the command, then the process must issue the H_GETCOM primitive.

The command may be either PRIVATE, meaning it can be issued only by the job submitter, or SYSTEM, meaning any operator may use it. This depends on the presence of the SYSTEM keyword.

- A system command is not deleted when the process that created it, is terminated (as is the case for private commands). So, it must be deleted by means of the H_DLCOM primitive.
- No message is issued when SYSTEM commands are created.
- System command names must be unique and different from already existing OCL commands.
- For a SYSTEM command, the process activated in the dispatching of the command must specify CONSOLE or USER parameter in the H-GETCOM primitive to get the full identification of the issuer and be able to send back messages onto the issuer's console.

In neither case is the command analyzed, that is, the text entered at the console is transferred directly from the OMH queue to the buffer given by the H_GETCOM primitive.

- For a SYSTEM command, the semaphore used must be declared by H_CSEMPOOL and got by H_CGETSEM.

2.3.2 H_DLCOM

Function

Deletes an operator command previously created by the H_CRCOM primitive.

Syntax

```
$H_DLCOM
          name ;
```

Parameters

name	Name of the command to be deleted (i_char4)

Normal return codes

DONE

Abnormal return codes

CDUNKN:	Command name unknown.

2.3.3 H_GETCOM

Function

Moves a command (input character string typed by operator) from the OMH queue to requestor's area.

Syntax

```
$H_GETCOM
          command
          ,IDENT = i_fb_15
          ,LENGTH = o_fb_15
          [,MAXLEN = i_fb_15]
          [,CONSOLE]
          [,USER = o_structure] ;
```

Parameters

command	o_structure. Reception zone for the command as typed in by the operator (or as elaborated by the analyzer).
IDENT	Identifier of the requested command. See "comments" under the H_CRCOM primitive.
LENGTH	Length of the reception zone for the command.
MAXLEN	Maximum length to be moved to the requestor's buffer. The default value is 64 characters, it must not exceed 192.
CONSOLE	If specified, this returns the operator number of the command submitter.

USER — Reception area for user/project/billing identity of the command submitter.

```
DCL 1   USER IDENT,
    2   USER-NAME       CHAR(12),
    2   USER-PROJECT    CHAR(12),
    2   USER-BILLING    CHAR(12) ;
```

The user-name field can be used by the USERID parameter of an H_SENDO primitive issued in relation to the same command. The project and billing are the user's default ones.

Normal return codes

DONE

Abnormal return codes

IDERR: — IDENT is invalid

Comments

The primitive moves the contents of the entry of the OMH queue into the requestor's buffer. The entire entry is moved if its length is less than or equal to MAXLEN. The entry is truncated if its length is greater than MAXLEN. The returned length is the length of the moved data. The right entry is retrieved by means of the IDENT field (REQID field of the semaphore message).

The output structure COMMAND has the following expansion:

1. If CONSOLE is specified:

```
DCL   1   COMMAND,
      2   *                BIT(32),      /*RFU  */
      2   OPERATOR         BIT(32),
      2   COMMAND_FIELDS   CHAR(N);
```

   COMMAND_FIELDS is exactly the text typed in from the console.

   LENGTH value is N + 8, where N is the number of characters typed in.

2. If CONSOLE is not specified:

   The COMMAND returned structure is only the text that was typed in. OPERATOR is not given back and the LENGTH value is N length of the command.

```
DCL  1  COMMAND,
     2  COMMAND_FIELDS    CHAR(N);
```

NOTE:

If the step using H_CRCOM and H_GETCOM belongs to a job launched by the system STARTUP, then all fields of the USER structure, if requested, are set to blanks even if USER, PROJECT and BILLING are specified in the JOB JCL statement.

2.3.4 H_OCL

Function

Sends the system a user command dynamically created by the H_CRCOM primitive.

Syntax

```
$H_OCL
          command
          ,LENGTH = i_fb15 ;
```

Parameters

command	i_char256: String containing the user command to send.
LENGTH	Length of the command string:

Normal Return code

DONE	H_OCL has successfully finished: The command has been correctly sent.

Abnormal Return codes

CDERR:	The command specified in the command string cannot be processed.
CDUNKN:	The command specified in the command string is unknown (not previously created by H_CRCOM or deleted by H_DLCOM).
SNDVIOL:	The submitting process is not authorized to use the command specified in the command string.

SPACEOV:	Allowed number of commands is exceeded for the submitting process-group.

Comments

The H_OCL primitive allows a GPL application to send to the system a user command and make it to react as if the command came from an operator at his console.

Although H_OCL may currently send commands belonging to the OCL domain, it is not recommended to use this primitive for these commands because the possibility will be suppressed in a further GCOS 7 release. For such commands use the GCL equivalent forms with the H_EXDIR primitive.

2.4 Task Management Primitives

2.4.1 H_NOTIFY

Function

Performs a V-operation on a semaphore with message. The message queuing is FIFO.

This primitive is mandatory for type 0 semaphores. For type 2 semaphores, the GPL built-in function SEVF may be used.

Syntax

```
$H_NOTIFY
          SEM = i_ptr
          ,REQID = i_bit_16
          ,MSG = i_bit_32;
```

Parameters

SEM	Pointer to the semaphore description of the semaphore on which a V-operation is to be performed
REQID	Request identifier returned at H_WAIT time
MSG	Message to be transmitted.

Return code

DONE

Comments

An abort may occur when the maximum semaphore count is exceeded.

2.4.2 H_RTNBPROCESS

Function

Returns the number of processes inside the running process group.

Syntax

```
$H_RTNBPROCESS
          name ;
```

Parameters

name	Name of the output variable (o_bit8); must be declared with the BYTE attribute.

Return codes

DONE

3. IOF Application SDI

The IOF Application SDI consists of the following set of IOF access primitives:

H_ASKIOF	sends and receives text from an interactive terminal..
H_CMDSYST	provides access to IOF directives and some system commands.
H_DCMODENV	declares the input structure for H_MODENV.
H_GETIOF	receives input text from an interactive terminal.
H_MODENV	requests modification of the environment.
H_PUTIOF	sends output text to an interactive terminal.

The GPL syntax conventions are summarized in the preface of this manual. For a complete description of GPL syntax and of GPL in general, please refer to the *GPL System Primitives Reference Manual*.

The IOF Application SDI has a specific MI (Marketing Identifier) which must be validated before program linkage.

Free-mode interface

These primitives provide a "free-mode" interface and allow you to activate some commands belonging to the IOF domain. They are all reserved for IOF interactive steps and processors.

With free mode, you can:

- exchange more than 256 characters with the terminal.
- bypass the "presentation" services: line folding, page lay-out, suppression of invalid characters.

This interface is needed by interactive applications working in screen mode, or with terminals not supported by the standard "presentation control".

Applications working in line mode must use the standard Terminal Access Method. The TAM record size limitation will be suppressed in a future version of GCOS 7.

Since the presentation services are not used, applications are responsible for the visibility of interactive dialog: screen clear, line switch, etc. The appropriate characters must be furnished in the message.

Applications also receive all characters sent by the terminal (for example, tabulation and cursor position).

NOTE:

Free mode is a special processing state:

- mixed usage, with calls to the Terminal Access Method, is forbidden.
- you should leave free mode when its features are no longer required.

Upon each transition (from normal mode to free mode, or from free mode to normal mode) on a screen, resynchronization takes place (+++ at the bottom of the screen) and the screen is cleared.

3.1 H_ASKIOF

Function

Asks the IOF user a question and requests an answer. (Reserved for IOF interactive steps and processors).

Syntax

```
$H_ASKIOF
          MESSAGE = i_ptr
          ,MESSL = i_fb15,
          ,REPLY = i_ptr
          ,REPLYL = o_fb15
          [,FORCE]
          [,HDR = o_structure];
```

Parameters

MESSAGE	Pointer to the message to be displayed.
MESSL	Length of the message to be displayed.
REPLY	Pointer to the user reply area.
REPLYL	Length of the reply entered at the terminal.
FORCE	Requires that the input comes from the terminal even if a Switch Input is active.

HDR

Must be a field 17 bytes long. This parameter contains the device header issued by the terminal. Note that the first byte is the length of the device header.

The device header field contains specific information issued by terminals such as VIP-synchronous, 3270, and 2780. Byte by byte, the most frequent formats are the following:

VIP:

```
00
03  STA  FC1  FC2
```

3270 family:

```
00
01  AID
03  AID  CA1  CA2
06  AID  CA1  CA2  SBA  A1  A2
```

2780 family:

The GPL structure to be used is shown below:

```
DCL 1 HDR,
    2 BSC1    BIT(1),
    2 BSC2    BIT(1),
    2 *       BIT(2),
    2 LENGTH  BIT(4),
    2 TEXT    CHAR(15);
```

For more information, refer to the specific terminal manual.

Normal return codes

DONE

Abnormal return codes

BREAK:	A break occurred during the I/O operation.
ADDRERR:	The reply area is not writeable or too short.
TERNAV:	Terminal not available.

Comments

In free mode:

- The output message length must be less than the length given by the H_DCMODENV/H_MODENV primitives; otherwise, truncation occurs.
- The output text must not have a prefix.
- The output area must be different from the reply area.

3.2 H_CMDSYST

Function

Enters the S_Y_S_T_E_M domain and provides access to IOF directives and to some system commands.

Syntax

```
$H_CMDSYST
          [COMMAND = i_char255, LENGTH = i_fb15]
          [,COMFILE = i_ifn]
          [,PREFIX = i_char1] ;
```

Parameters

COMMAND	Specifies the command string that contains the command(s) to be executed. The COMMAND and COMFILE parameters are mutually exclusive. If multiple commands are stored in the command string, they must be separated by semicolons.
LENGTH	Length (in bytes) of the command string. This parameter is mandatory if COMMAND is used.
COMFILE	Specifies the internal file name (declared by H_FD) of the sequential input file or subfile that contains the commands to be executed. Before calling H_CMDSYST, COMFILE must be assigned, but not opened, and be given the DATAFORM=SSF and TRUNCSSF attributes. The COMMAND and COMFILE parameters are mutually exclusive.
PREFIX	Specifies the prompt character used to request input commands or IOF directives from the terminal. Default value is S.

Return Codes

The return code of the last command processed in the S_Y_S_T_E_M domain is returned.

Comments

- H_CMDSYST allows you to enter the S_Y_S_T_E_M domain. Within this domain, IOF directives and some system commands can be executed:
  - if the COMMAND parameter is used, the S_Y_S_T_E_M domain is entered and then IOF directives or commands specified in the command string are executed. The S_Y_S_T_E_M domain is exited when either the end of the command string has been reached or a QUIT (or /) command from the command string has been executed.
  - If the COMFILE parameter is used, the S_Y_S_T_E_M domain is entered and then IOF directives or commands stored in the COMFILE file are executed. The S_Y_S_T_E_M domain is exited when either the end of the COMFILE file has been reached or a QUIT (or /) command from the COMFILE file has been executed.
  - If both COMMAND and COMFILE are omitted, the S_Y_S_T_E_M domain is entered, the prompt character specified by the PREFIX parameter is printed, and the terminal is switched to input mode, waiting for IOF directives or system commands to be entered. The S_Y_S_T_E_M domain is exited when a QUIT (or /) command has been given and executed.
- Within the S_Y_S_T_E_M domain, the following GCL commands are available:

`BUILD_FILE`	(allocates a disk file)
`BUILD_LIBRARY`	(builds a library)
`CREATE_DIR`	(creates a directory or master directory)
`DELETE_DIR`	(deletes a directory or master directory)
`DELETE_FILE`	(deletes a file)
`DELETE_LIBRARY`	(deletes a library)
`EDIT`	(enters the Text Editor)
`FSE`	(enters the Full Screen Editor)
`LIST_DIR`	(lists a directory and its dependent objects)
`LIST_FILE`	(lists file information)
`LOAD_FILE`	(loads a file)
`MAINTAIN_LIBRARY`	(enters Library Maintenance)
`QUIT`	(leaves the S_Y_S_T_E_M domain)

- The normal use of H_CMDSYST is in interactive mode. This primitive may also be used in batch mode, but in this case either the COMMAND or COMFILE parameter must be specified. H_CMDSYST must not be used under TDS.

- In order to reserve resources needed by H_CMDSYST, the following options must be used when linking the load module:

```
USEDSM=H_SM9,
STARTASG=(PRIVATE=11),
VACSEG=(SHARE=+120),
STACK1=(INITSIZE=10K,MAXSIZE=64K),
STACK3=(INITSIZE=8K,MAXSIZE=64K),
```

  In addition, three type-3 vacant entries in STN 1, 2, or 3 must be reserved; for example:

```
SEGTAB3=(SHRLEVEL=3,VSEG=3)
```

Examples

1. Enter the S_Y_S_T_E_M domain and wait for commands coming from the terminal:

```
$H_CMDSYST;
```

2. Process the LSDIR command:

```
DCL 1  COMMAND_STRUCT,
    2  COMMAND_TEXT    CHAR(255),
    2  COMMAND_LENGTH  FIXED BIN(15);

command_text = "LSDIR";
command_length = 5;

$H_CMDSYST  COMMAND = command_text,
            LENGTH = command_length;
```

3. Process the BLIB and LSDIR commands:

```
DCL 1  command_struct,
    2  command_text    char(255),
    2  command_length  fixed bin(15);

command_text = "BLIB DUAL.MEM.FILX$RES SIZE=1
              MEMBERS=10;LSDIR";
command_length = 46;

$H_CMDSYST   COMMAND = command_text
             LENGTH = command_length;
```

4. Process commands stored in the subfile EXEC_COM of file ORDERS:

```
$H_FD
         ifile
         TRUNCSSF
         DATAFORM = SSF;
$H_ASSIGN
         ifile
         EFN = "ORDERS"
         SUBFILE= "EXEC_COM"
         POOL = "0"B
         FILESTAT= "00"X
         SHARE="10"X ;
$H_CMDSYST
         COMFILE = ifile;
```

The subfile specified as 'comfile' (EXEC_COM in this example) may contain any commands belonging to the S_Y_S_T_E_M domain. For example:

```
LSDIR DUAL.MEM;
BLIB DUAL.MEM.FILX$RES SIZE=1 MEMBERS=10;
LSF DUAL.MEM.FILX$RES ALL;
```

3.3 H_DCMODENV

Function

Declares the input structure used by the H_MODENV primitive. (Use reserved for IOF interactive steps and processors).

This allows you to modify parameters of the IOF environment according to a control mask (fields named "_C" in the structure). Each field to be modified must be selected (bit set to 1) in the control mask; others must be set to 0.

Syntax

```
$H_DCMODENV
          [PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1] ;
```

Parameters

PREFIX	Prefix value.
ATTRIB	GPL attribute for the structure.
NLVL1	If NLVL1 is specified, the characters "DCL" and the end-of-structure semicolon are not generated. The structure starts at level 3.

Expansion

```
/*__..__  $H_DCMODENV; */

DCL 1 MODENV,
    2 CONTROL_MASK,                    /* PARAMETERS TO BE TAKEN INTO ACCOUNT      */
    3 FREEMODE_C           BIT(1),     /* FREEMODE                                 */
    3 BREAK_HANDLING_C     BIT(1),     /* BREAK HANDLING                           */
    3 QUIT_C               BIT(1),     /* QUIT UPON A BREAK                        */
    3 RECONNECTION_C       BIT(1),     /* RECONNECTION STRATEGY                    */
    3 INPREFIX_C           BIT(1),     /* DEFAULT INPUT PREFIX                     */
    3 RSU1_C               BIT(2),     /* RESERVED FOR SOFTWARE USE                */
    3 RFU_C                BIT(25),    /* RESERVED FOR FUTURE USE                  */
    2 CONTROL_RFU          CHAR(12),   /*                                          */
    2 VALUE_FLAGS,                     /*                                          */
    3 H_FREEMODE_V,                    /*                                          */
    4 FREEMODE_TRANSITION  BIT(1),     /* 0 : FREEMODE EXITED                      */
                                       /* 1 : FREEMODE ENTERED                     */
                                       /* WALG MUST BE SPECIFIED                   */
    4 FREEMODE_MSG         BIT(1),     /* INDICATES IF THE MESSAGES                */
                                       /* COMING THROUGH THE MAILBOX               */
                                       /* MUST BE DISPLAYED                        */
    3 RECONNECTION_V       BIT(1),     /* CONCERNS THE RECONNECTION                */
                                       /* FEATURE IN CASE OF TERMINAL FAILURE      */
                                       /* 1 : MECHANISM ACTIVATED                  */
                                       /* 0 : STEP WILL BE IN ALL CASES ABORTED    */
    3 QUIT_V               BIT(1),     /* IF 1, BRK HANDLER CALLED AT              */
                                       /* QUIT UPON BREAK                          */
    3 RFU_VFLG             BIT(28),    /* RESERVED FOR FUTURE USE                  */
    2 VALUE_FIELDS,                    /*                                          */
    3 FREEMODE_WALG        FIXED BIN(15), /* LIMIT SIZE OF PUT/GET/ASKIOF          */
                                       /* PRIMITIVES IN FREEMODE                   */
    3 BREAK_OPTION         BIT(8),     /* 00x:STANDARD BREAK HANDLING              */
                                       /* 01x:SPECIAL BREAK HANDLING               */
    3 BREAK_PROC           ENTRY,      /* USERS BREAK ROUTINE                      */
    3 INPREFIX_V           CHAR(3),    /* NEW DEFAULT PREFIX TO BE                 */
                                       /* USED BY GETIOF                           */
    3 RSU3_V               CHAR(15),   /* RESERVED FOR SOFTWARE USE                */
    2 VALUE_RFU            CHAR(51);   /*                                          */
```

FREEMODE_C

Controls operations relative to free mode.

When FREEMODE_TRANSITION="0"B, free mode is exited. In this case FREEMODE_MSG and FREEMODE_WALG are meaningless.

When FREEMODE_TRANSITION="1"B, free mode is entered. In this case FREEMODE_MSG indicates whether the messages contained in the mailbox must or must not be displayed along with the primitives issued in free mode. The field FREEMODE_WALG defines the limit size for get/put/ask IOF primitives.

In free mode, all IOF presentation services (for example, suppression of trailing blanks, line folding, character substitution) are bypassed.

BREAK_HANDLING_C

Controls break handling.

If BREAK_HANDLING_C="1"B, then the BREAK_OPTION and BREAK_PROC fields are significant.

When BREAK_OPTION is given the value "00"X, standard break handling is activated. In this case, the break routine specified by the BREAK_PROC entry is called only after the IOF standard break treatment and only if required by the terminal user (interrupt, /, end of current action commands).

When BREAK_OPTION is given the value "01"X, the break routine specified by the BREAK_PROC entry is called directly upon a break action.

The BREAK_PROC entry defines the break routine to be called. This entry must be given either a valid procedure entry or a NULL() value (hexadecimal FFFFFFFF).

QUIT_C

Controls the action to be taken when a break occurs and a QUIT command is entered.

When QUIT_V is given the value "1"B, if a break occurs and a QUIT command is entered, the current step is not aborted and the break routine specified by the BREAK_PROC entry is called.

RECONNECTION_C

Controls how the terminal reconnection feature is used.

If RECONNECTION_V="1"B, the reconnection mechanism will be activated normally.

If RECONNECTION_V="0"B, the current step will be aborted in the case of a terminal failure. For example, the step might hold vital resources and not want to retain them over a long period of time.

INPREFIX_C

INPREFIX_V is the default input prefix to be used in user steps or processors.

3.4 H_GETIOF

Function

Gets input text from a terminal. (Use reserved for IOF interactive steps and processors.)

Syntax

```
$H_GETIOF
          REPLY = i_ptr
          [,PREFIX = i_char3]
          ,LENGTH = o_fb15
          [,FORCE]
          [,NPREFIX]
          [,HDR = o_structure] ;
```

Parameters

REPLY	Pointer to the working area where the input text will be sent.
PREFIX	Prefix sent to the terminal to request input text (prompt message). Default = "I:". This default value can be changed by the H_MODENV primitive.
LENGTH	Length of the text entered at the terminal.
FORCE	Requires that the input comes from the terminal even if a Switch Input is active.
NPREFIX	When specified, no prompt is sent.

HDR

Must be a field 17 bytes long. This parameter contains the device header issued by the terminal. Note that the first byte is the length of the device header.

The device header field contains specific information issued by terminals such as VIP-synchronous, 3270, and 2780. Byte by byte, the most frequent formats are the following:

VIP:

```
00
03  STA  FC1  FC2
```

3270 family:

```
00
01  AID
03  AID  CA1  CA2
06  AID  CA1  CA2  SBA  A1  A2
```

2780 family:

The GPL structure to be used is shown below:

```
DCL 1 HDR,
    2 BSC1    BIT(1),
    2 BSC2    BIT(1),
    2 *       BIT(2),
    2 LENGTH  BIT(4),
    2 TEXT    CHAR(15);
```

For more information, refer to the specific terminal manual.

Normal return codes

DONE

Abnormal return codes

BREAK:	A break occurred during the I/O operation.
ADDRERR:	The reply area is not writeable or too short.
TERNAV:	Terminal not available.

Comments

- This primitive enables the user to get input text from a terminal or from a source library member (if a Switch Input is active).

  If the FORCE parameter is specified, however, data input will be prompted at the terminal; any active Switch Input will be ignored. After a primitive with FORCE has been executed, any active (but ignored) Switch Input will remain active.

- If a break is issued, the returned length is zero.

3.5 H_MODENV

Function

Allows an interactive program to modify some characteristics of the environment in which it executes. (Use reserved for IOF interactive steps and processors).

Syntax

```
$H_MODENV

            i_structure;
```

Parameters

As declared by the H_DCMODENV primitive.

Normal return codes

DONE

Abnormal return codes

SNDARERR:	Error in an input field.

Comments

This primitive allows an interactive application to modify some characteristics of its environmental parameters, according to a series of control flags.

The flags set to 1 in the control mask show the parameters that the caller wants to modify. The flags left set to 0 in the control mask indicate that the corresponding parameters values are not to be changed.

For each flag set to 1 in the control mask, a corresponding field value exists within the input structure.

3.6 H_PUTIOF

Function

Sends output text to the terminal. (Use reserved for IOF interactive steps and processors).

Syntax

```
$H_PUTIOF
          MESSAGE = i_ptr
          ,LENGTH = i_fb15 ;
```

Parameters

MESSAGE	Pointer to the message to be output.
LENGTH	Length of the text to be output.

Normal return codes

DONE

Abnormal return codes

BREAK:	A break occurred during the I/O operation.
TERNAV:	Terminal not available.

Comments

This primitive enables the user to output messages on the terminal.

In free mode:

- No prefix is added to the output text.
- The length of the message to be output must be less than the length given by the H_DCMODENV/H_MODENV primitives; otherwise, truncation occurs.

4. File Management Tools SDI

The File Management Tools SDI consists of the following set of file, volume, and device access primitives:

Access to VTOC:

H_READVTOC	reads the VTOC of a FBO or VBO disk
H_DCEXTDESC	declares an output structure for file or volume extents
H_DCEXTVOL	declares an output structure for the volume label
H_DCLOGLAB	declares an output structure for the logical label
H_DFL1	declares an output structure for file label 1
H_DFL2	declares an output structure for file label 2
H_DFL3	declares an output structure for file label 3
H_DFL5	declares an output structure for file label 5

Access to extended UFAS labels:

H_FILINFO	gets information from the UFAS label of a UFAS file
H_DCFILINFO	declares an output structure for H_FILINFO

Access to disk type:

H_DCMSINFO	declares an output structure for H_MSINFO
H_DCMSINFO_MIRROR	declares an output structure for H_MSINFO to contain mirrored disk information
H_MSINFO	gets information on a specified media

Access to device configuration and information:

H_DVLIST	lists the devices of a given type connected to a system
H_DCDEVINFO	declares an output structure for H_DEVINFO
H_DEVINFO	returns information about a specified device

Access to file size:

H_GTSIZE	returns the size and increment for a file or library
H_DCGTSIZE	declares an output structure for H_GTSIZE

Access to file and volume utilities:

H_COPY_FILE	copies an input file to an output file of the same type
H_DCENDMIG	declares the H_ENDMIG optional input parameters
H_DCFMIG	declares the contents of an OUTFILE file record generated by the MIGOUT command
H_ENDMIG	performs complementary GCOS 7 functions after a save server operation
H_LOAD_FILE	loads a library member or UFAS file, reorganizes a UFAS file
H_PRINT_FILE	prints records from a file or a library member
H_RESTORE_FILE	restores a disk file from a sequential UFAS disk/tape file
H_SAVE_FILE	saves a disk file into a UFAS sequential disk/tape file
H_SAVE_DISK	saves a disk volume into a sequential UFAS disk/tape file

Assigned ifns:

H_IFNASG	lists the ifns assigned to a step

Cartridge volume information:

H_DCVOLINFO	declares an output structure for H_GETVOLINFO
H_GETVOLINFO	gets information about a cartridge volume

Modification of a file's catalog description:

H_MODIFY	modifies a file or generation description in a catalog

The GPL syntax conventions are summarized in the preface of this manual. For a complete description of GPL syntax and of GPL in general, please refer to the *GPL System Primitives Reference Manual*.

The File Manager SDI set of primitives has a specific MI (Marketing Identifier) which must be validated before program linkage.

4.1 File-level Primitives

4.1.1 H_COPY_FILE

Function

Copies the contents of an input file to an output file of identical type.

Syntax

```
$H_COPY_FILE
          infile
          ,outfile
          [,OUTALC = i_bit1]
          [,CATNOW = i_bit1]
          [,FMEDIA = i_bit1]
          [,UPDJRNL = i_bit1]
          [,NWRJOR = i_bit1] ;
```

Parameters

infile	specifies the internal file name (declared by H_FD) of the input file that contains the data to be copied.
outfile	specifies the internal file name (declared by H_FD) of the output file.
OUTALC	indicates whether or not the missing output file is to be dynamically allocated taking the input file as model. "1"B: dynamically allocates the missing output file before the copy takes place. Note that if an output file with the same characteristics already exists

	when H_COPY_FILE is issued and this file is too small, then this file will be extended in order to contain the whole input file. "0"B: (default) uses the existing output file.
CATNOW	indicates whether or not the output file is to be cataloged during dynamic allocation. "1"B: the output file is to be cataloged during dynamic allocation. OUTALC must be set to "1"B, and FILESTAT="04"X (UNCAT) must be specified for the output file in H_ASSIGN. "0"B: (default) the output file is not to be cataloged during dynamic allocation.
FMEDIA	for a cataloged output file, specifies whether or not the existing cataloged list of volumes is to be erased and replaced by the volumes (given in H_ASSIGN) onto which dynamic allocation took place. "1"B: replace in the catalog, for the output file, the existing list of volumes. OUTALC="1"B and CATNOW="1"B must also be specified. "0"B: (default) do not replace in the catalog, for the output file, the list of volumes. The file level catalog entry must be empty.
UPDJRNL	specifies if H_COPY_FILE has to interface with the After Journal in order to reset to empty the set of after images for the input file since its previous save by H_SAVE_FILE, H_FILSAVE, SAVE_FILE(SET) or H_COPY_FILE with UPDJRNL="1"B. "1"B: reset to empty after images. "0"B: (default) do not reset to empty after images.

NWRJOR	indicates whether or not H_COPY_FILE may write messages to the Job Occurrence Report (JOR). "1"B: no message is written to the JOR. "0"B: (default) messages are written to the JOR.

Normal return codes

DONE	H_COPY_FILE is successfully done. Outfile is loaded.
DONEID	H_COPY_FILE is successfully completed but the user specified that irrecoverable read errors were to be skipped over (ERROPT=SKIP in the infile H_FD). Such errors were encountered and several parts of infile have not been copied; the integrity and consistency of outfile should not be assumed.

Abnormal return codes

ARGERR	one (or more) given parameter(s) is (are) wrong (not identical device classes, logical characteristics of infile and outfile are different, DUMMY file assigned for either infile or outfile, etc...).
ARVIOL	access rights violation. The user does not have the required access rights on either the input or the output file.
CATERR	assignment of either the input file or the output file contradicts their actual cataloged or uncataloged file status.
DATANAV	the input file is in an unstable state.
DVNAV	there are not enough devices (disk or tape drives) currently available to meet the request. The DEFER option was used for a file assignment.
EFNUNKN	either the input file or the output file does not exist on the list of volumes retrieved from H_ASSIGN. OUTALC="1"B may be used for a missing output file.

EXHAUST	the end of outfile has been reached and OUTALC="1"B has not been specified for extension or there is no more free space on the media list that supports outfile.
IFNERR	either the infile or the outfile parameter is wrong; leading to invalid control structures.
IFNNASG	one of the files, input or output is not assigned. Both files should be assigned.
IOFAIL	an irrecoverable read error was encountered either on the input file or on the output file.
MDNAV	a volume mounting request has been canceled by the operator.
MDERR	incorrect or partial media list given for infile or outfile.
NOTOBS	the outfile tape to be overwritten is still valid, the expiry date has not been reached.
NOWAIT	synchronization error. Infile is being accessed by another task.
OPEN	at least one of the files, infile or outfile is open. All these files should be closed before H_COPY_FILE is issued.
TRUNC	refer to the EXHAUST return code.
SPACENAV	there is not enough available space on the list of volumes specified in H_ASSIGN to allocate the output file.
YETALLOC	the catalog entry referenced by the output file already contains a list of volumes. FMEDIA="1"B must be used to continue.

Some other return codes may also be returned to the caller especially when OUTALC="1"B is specified to request dynamic allocation of the missing output file (refer to H_PREALLOC in the *GPL System Primitives Reference Manual*).

Comments

- H_COPY_FILE copies an input file (as referenced by the infile parameter) into an output file (as referenced by the outfile parameter) of identical file attributes (format, organization, CISIZE, BLKSIZE, etc..).
- Device classes on which both infile and outfile reside must also be identical. Note that CT and MT are considered as identical device classes but MS/B10 VBO and MS/B10 FBO are not considered as identical device classes.
- Infile and outfile must be assigned and must be closed when H_COPY_FILE is called.
- To get a cataloged outfile, both CATNOW="1"B and OUTALC="1"B must be specified and FILESTAT="04"X (UNCAT) must be specified in H_ASSIGN. When the H_COPY_FILE operation is over, the outfile's status changes from "04"X (UNCAT) to "00"X (CAT).
- Whenever a malfunction occurs during the execution of an H_COPY_FILE operation, the malfunction is reported to the user via JOR messages (except when NWRJOR="1"B is specified) and via return codes. The step completion code is then forced to SEV1 regardless of its previous value.

For more information, refer to the FILDUPLI utility in the *Data Management Utilities User's Guide*.

4.1.2 H_DCENDMIG

Function

H_DCENDMIG is used to declare the H_ENDMIG optional input parameters.

Syntax

```
$H_DCENDMIG     [TYPE = {MIGOUT | MIGIN}]
                [,PREFIX = l_identifier16]
                [,ATTRIB = l_char]
                [,{INIT | NINIT}]
                [,NLVL1 [LEVEL =i_cst]]
                ;
```

Parameters

TYPE	Specifies the current migration operation type. MIGOUT: migration-out operation (default); MIGIN: migration-in operation.
PREFIX	Specifies a character string to be used as a prefix for the name of the structure and of each elementary item. (Default is none).
ATTRIB	Specifies the attributes of the structure. (Default is none).
INIT	Initialization of the structure. (INIT is the default except when the structure has BASED or DEFINED attributes).
NINIT	The structure initialization is not performed.
NLVL1	Requires not to generate the structure level 1. (By default level 1 is generated).
LEVEL	Depth for structure fields (to be used with NLVL1). (Default value is 2).

Expansion

The generated structure contents depend on the migration operation type.

The primitive $H_DCENDMIG TYPE=MIGOUT, PREFIX=MIGOUT_; produces the following structure:

```
/* $H_DCENDMIG TYPE=MIGOUT, PREFIX=MIGOUT_; */
DCL 1 MIGOUT_ENDMIG,                              /* input parameters for H_ENDMIG        */
                                                  /* TYPE = MIGOUT                        */
     2 MIGOUT_VERSION    FIXED BIN(15) INIT(0),   /* version (first=0)                    */
     2 MIGOUT_SERVICE    BIT(8) INIT("01"X),      /* ="00"X default archive tool on       */
                                                  /* the site                             */
                                                  /* ="01"X user solution                 */
     2 *                 CHAR(1) INIT("00"H),     /* rfu                                  */
     2 MIGOUT_FILE_GROUP CHAR(64) INIT((64)" "),  /* file group name for save             */
     2 MIGOUT_SAVE_DATE,                          /* file archive date                    */
                                                  /* format is INTERNAL                   */
                                                  /* (see H_TIME and H_TRTIME).           */
      3 MIGOUT_TIME      FIXED BIN(31) INIT(0),   /* time of day                          */
                                                  /* (unit = milli-second)                */
      3 MIGOUT_DATE,
       4 MIGOUT_YR       FIXED BIN(15) INIT(0),   /* year                                 */
       4 MIGOUT_DD       FIXED BIN(15) INIT(0),   /* day in year                          */
     2 MIGOUT_EXPIRATION_DATE,                    /* save expiration date (internal       */
                                                  /* format see H_TIME and H_TRTIME)      */
      3 MIGOUT_EXPTIME   FIXED BIN(31) INIT(0),   /* time of the day (milli-second)       */
      3 MIGOUT_EXPDATE,
       4 MIGOUT_EXPYR    FIXED BIN(15) INIT(0),   /* year                                 */
       4 MIGOUT_EXPDD    FIXED BIN(15) INIT(0),   /* day in year                          */
     2 MIGOUT_OPTIONS,                            /* options for migration-out            */
      3 MIGOUT_FORCE     BIT(1) INIT("0"B),       /* =1 the file must be deallocated      */
                                                  /* without pre-migration step.          */
                                                  /* =0 FORCE not specified               */
      3 MIGOUT_PREMIG    BIT(1) INIT("0"B),       /* =1 the file deallocation is          */
                                                  /* deferred (pre-migration step),       */
                                                  /* =0 PREMIG not specified              */
      3 *                BIT(6) INIT((6)"0"B),    /* rfu                                  */
     2 *                 CHAR(171) INIT((171)"00"H); /* rfu                               */
```

The primitive $H_DCENDMIG TYPE=MIGIN, PREFIX=MIGIN_; produces the following structure:

```
/* $H_DCENDMIG TYPE=MIGIN, PREFIX=MIGIN_; */
DCL 1 MIGIN_ENDMIG,                              /* input parameters for H_ENDMIG        */
                                                 /* TYPE = MIGIN                         */
     2 MIGIN_VERSION     FIXED BIN(15) INIT(0),  /* version (first=0)                    */
     2 MIGIN_SERVICE     BIT(8) INIT("01"X),     /* ="00"X default archive tool on       */
                                                 /* the site                             */
                                                 /* ="01"X user solution                 */
     2 MIGIN_ABNTERM_OPTIONS,                    /* options for migration-in             */
                                                 /* operations abnormal termination      */
      3 MIGIN_CLEAN      BIT(1) INIT("0"B),      /* =1 the file must be deallocated
                                                  from the disks indicated in the
                                                  catalog description
                                                  (e.g: IOFAIL on restore)             */
      3 MIGIN_NLAST      BIT(1) INIT("0"B),      /* =1 the jobs waiting for the
                                                  migration-in completion must not be
                                                  released now since it is not the
                                                  last attempt to restore this file    */
      3 *                BIT(6) INIT((6)"0"B),   /* rfu                                  */
     2 *                 CHAR(252) INIT((252)"00"H); /* rfu                              */
```

4.1.3 H_DCFILINFO

Function

Declares the output structure for use by the H_FILINFO primitive.

Syntax

```
$H_DCFILINFO

          [,PREFIX = l_identifier10]
          [,ATTRIB = l_char]
          [,NLVL1] ;
```

Parameters

PREFIX	Prefix of the declarative structure variables.
ATTRIB	Used to qualify the declarative structure.
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Return codes

NONE

Expansion

```
/*__..__  $H_DCFILINFO;                                                    */

DCL 1  INFO_SP1,                          /* LABEL RETURNED INFORMATION (SPACE1)      */
    2  DCFILINFO_VERSION        BIT (8),  /* MACRO VERSION NUMBER "01"X: V1, V2, V3   */
    2  FIL_INFO,                         /* GENERAL INFORMATION ABOUT FILE           */
    3  FL_CREAT_SYSTEM_WIDTH  BIT (8),   /* UFAS FILE CREATION RELEASE "01"X:        */
                                         /* V1, V2, V3 NLS "02"X: V3 LS              */
    3  FILE_NAME              CHAR(44),  /* EXTERNAL FILE NAME                       */
    3  ZONE_LOGIC_MS,                    /* FILE ORGANIZATION ZONE                   */
    4  LOGIC_CATAL          BIT (8),     /* CATALOG: YES (1) ! NO (2)                */
    4  *                    BIT (16),    /* RFU                                      */
    4  LOGIC_BLK_CISIZE     FIXED BIN(15),
    4  LOGIC_BLK_CISZ_PM    CHAR(3),
    4  LOGIC_RECFORM        CHAR(2),     /* RECFORM: FB, VB                          */
    4  LOGIC_INCRSIZE       BIT (24),    /* INCRSIZE:                                */
    4  LOGIC_UNIT           BIT (8),     /* UNIT: (1)ATR (2)BLK (3)TRK (4)CYL        */
    4  LOGIC_ORGANIZATION   BIT (8),     /* LOGIC_ORGANIZATION < 6                   */
                                         /* or LOGIC_ORGANIZATION > 10               */
                                         /* RESERVED  6 ==> UFAS SEQ                 */
                                         /*           7 ==> UFAS RELATIVE            */
                                         /*           8 ==> UFAS INDEXED             */
                                         /*           9 ==> UFAS IDS (EX RANDOM)     */
                                         /* UFAS SANS LABEL ETENDU ==>               */
                                         /* LOGIC_ORGANIZATION=9 for IDS             */
                                         /* LOGIC_ORGANIZATION=6 for other UFAS      */
    4  LOGIC_RECSIZE          FIXED BIN(15),  /* RECSIZE                            */
    4  LOGIC_RECSIZE_PM       CHAR(3),
    4  *                      CHAR(2),        /* RFU                                */
    4  LOGIC_MAXSIZE          FIXED BIN (31), /* MAXSIZE                            */
    4  *                      CHAR(5),        /* RFU                                */
    4  LOGIC_COMPFILE         BIT (8) ,       /* NOT COMPACT FILE ("02"X)           */
    4  *                      BIT (16),       /* RFU                                */
    4  LOGIC_IDXTYPE          BIT (8),        /* IDXTYPE: 1 SPLIT                   */
                                              /*          2 GLOBAL                  */
                                              /*          3 SECONDRY                */
                                              /*          4 WORK                    */
    4  LOGIC_CASIZE           FIXED BIN(15),  /* CASIZE                             */
    4  LOGIC_CIFSP            FIXED BIN(15),  /* CIFSP                              */
    4  LOGIC_CAFSP            FIXED BIN(15),  /* CAFSP                              */
    4  *                      CHAR (1),       /* RFU                                */
    4  LOGIC_TURBO_CISIZE     FIXED BIN(31),  /* CISIZE POUR UFAS TURBO             */
    4  LOGIC_TURBO_RECSIZE    FIXED BIN(31),  /* RECSIZE POUR UFAS TURBO            */
    4  *                      CHAR(16),       /* RFU                                */
    3  ZONE_USAGE_MS,                        /* FILE USAGE ZONE                    */
    4  USAGE_CREDATE,                        /* CREATION DATE  YEAR/DAY            */
    5  USAGE_CREA_YEAR        BIT(8),         /* CREATION YEAR                      */
    5  USAGE_CREA_DAY         BIT(16),        /* CREATION DAY                       */
    4  USAGE_EXPDATE,                        /* EXPIRATION DATE  YEAR/DAY          */
    5  USAGE_EXP_YEAR         BIT(8),         /* EXPIRATION YEAR                    */
    5  USAGE_EXP_DAY          BIT(16),        /* EXPIRATION DAY                     */
    4  USAGE_GENSYS           CHAR(13),       /* GENERATING SYSTEM                  */
    4  USAGE_VSN              CHAR(6),        /* VSN FIRST VOLUME                   */
    4  USAGE_VSEQN            FIXED BIN(15),  /* VSEQN OF THIS VOLUME               */
    4  USAGE_STATUS_TXT       CHAR(8),        /* FILE STATUS TEXT                   */
    4  USAGE_CI_FORMAT        CHAR(1),        /* DATA CI FORMAT                     */
    4  USAGE_NUMLAB           FIXED BIN(15),  /* #USER'S LABELS                     */
    4  USAGE_NUMEXT           FIXED BIN(15),  /* # OF EXTENDS THE FILE LIES ON, IN THE */
                                              /* FIRST VOLUME. THE USER LABELS EXTENTS */
                                              /* ARE NOT INCLUDED                   */
    4  USAGE_FILVERS          BIT (8),        /* FILE VERSION NUM.                  */
    4  *                      CHAR(16),       /* RFU                                */
```

```
3  ZONE_STAT_LOGGING,                    /* STATISTICAL LOGGING INFORMATION          */
4  LOG_ACTIVEDATACI       BIT(24),       /* # ACTIVE DATA CIs IN FILE                */
4  LOG_CISPLITTINGS       BIT(16),       /* # CI SPLITTINGS                          */
4  LOG_CASPLITTINGS       BIT(16),       /* # CA SPLITTINGS                          */
4  LOG_LASTSPLITTEDCI     BIT(24),       /* LAST SPLITTED CI                         */
4  LOG_LASTSPTDADDRSP     BIT(8),        /* LAST SPLITTED ADDR SPACE                 */
4  LOG_ACTIVESECTION      BIT(24),       /* # ACTIVE SECTIONS                        */
4  LOG_CISPLTSECTION      BIT(16),       /* # CI SPLIT SECTIONS                      */
4  LOG_CASPLTSECTION      BIT(16),       /* # CA SPLIT SECTIONS                      */
2  UF_INFO2,                             /* ADDRESS SPACE INFO FOR PRIMARY INDEX     */
3  DATAS,                                /* ADDRESS SPACE INFO FOR DATAS             */
4  *                      CHAR(2),       /* CHARACTERS "S2"                          */
4  PRIDX_S2ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S2)                       */
4  PRIDX_S2FORMT          FIXED BIN(31), /* #FORMATTED CI (S2)                       */
4  PRIDX_S2CITRK          FIXED BIN(15), /* #CI PER TRACK (S2)                       */
4  PRIDX_S2CISZ           FIXED BIN(31), /* CI SIZE    (S2)                          */
4  PRIDX_S2EXT            FIXED BIN(15), /* #EXTENT    (S2)                          */
3  PRIMARY_HIGHEST,                      /* ADDRESS SPACE INFO FOR PRIMARY HIGHEST   */
4  *                      CHAR(2),       /* CHARACTERS "S3"                          */
4  PRIDX_S3ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S3)                       */
4  PRIDX_S3FORMT          FIXED BIN(31), /* #FORMATTED CI (S3)                       */
4  PRIDX_S3CITRK          FIXED BIN(15), /* #CI PER TRACK (S3)                       */
4  PRIDX_S3CISZ           FIXED BIN(31), /* CI SIZE   (S3)                           */
4  PRIDX_S3EXT            FIXED BIN(15), /* #EXTENT    (S3)                          */
3  PRIMARY_LOWEST,                       /* ADDRESS SPACE INFO FOR PRIMARY LOWEST    */
4  *                      CHAR(2),       /* CHARACTERS "S4"                          */
4  PRIDX_S4ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S4)                       */
4  PRIDX_S4FORMT          FIXED BIN(31), /* #FORMATTED CI (S4)                       */
4  PRIDX_S4CITRK          FIXED BIN(15), /* #CI PER TRACK (S4)                       */
4  PRIDX_S4CISZ           FIXED BIN(31), /* CI SIZE    (S4)                          */
4  PRIDX_S4EXT            FIXED BIN(15), /* #EXTENT    (S4)                          */
2  UF_INFO3,                             /* ADDRESS SPACE INFO FOR SECONDARY INDEXES */
3  SECONDARY_HIGHEST,                    /* ADDRESS SPACE INFO FOR SECONDARY HIGHEST */
4  *                      CHAR(2),       /* CHARACTERS "S6"                          */
4  SCIDX_S6ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S6)                       */
4  SCIDX_S6FORMT          FIXED BIN(31), /* #FORMATTED CI (S6)                       */
4  SCIDX_S6CITRK          FIXED BIN(15), /* #CI PER TRACK (S6)                       */
4  SCIDX_S6CISZ           FIXED BIN(31), /* CI SIZE    (S6)                          */
4  SCIDX_S6EXT            FIXED BIN(15), /* #EXTENT    (S6)                          */
3  SECONDARY_LOWEST,                     /* ADDRESS SPACE INFO FOR SECONDARY LOWEST  */
4  *                      CHAR(2),       /* CHARACTERS "S7"                          */
4  SCIDX_S7ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S7)                       */
4  SCIDX_S7FORMT          FIXED BIN(31), /* #FORMATTED CI (S7)                       */
4  SCIDX_S7CITRK          FIXED BIN(15), /* #CI PER TRACK (S7)                       */
4  SCIDX_S7CISZ           FIXED BIN(31), /* CI SIZE    (S7)                          */
4  SCIDX_S7EXT            FIXED BIN(15), /* #EXTENT    (S7)                          */
3  SECONDARY_DENSE,                      /* ADDRESS SPACE INFO FOR SECONDARY DENSE   */
4  *                      CHAR(2),       /* CHARACTERS "S5"                          */
4  SCIDX_S5ALLCI          FIXED BIN(31), /* #ALLOCATED CI (S5)                       */
4  SCIDX_S5FORMT          FIXED BIN(31), /* #FORMATTED CI (S5)                       */
4  SCIDX_S5CITRK          FIXED BIN(15), /* #CI PER TRACK (S5)                       */
4  SCIDX_S5CISZ           FIXED BIN(31), /* CI SIZE    (S5)                          */
4  SCIDX_S5EXT            FIXED BIN(15), /* #EXTENT    (S5)                          */
2  KEY,                                  /* KEYS ZONE                                */
3  KEY_NUMKEY             FIXED BIN(15), /* NUMBER OF SECONDARY KEYS                 */
3  KEY_COLLATE            BIT(8),        /* COLLATE: 1 EBCDIC                        */
                                         /*          2 ASCII                         */
                                         /*          3 BCD                           */
3  KEY_INFO               (16),          /* PRIMARY KEY (ENTRY 1) AND SECONDARY      */
                                         /* KEYS (ENTRY 2 TO 16) INFO                */
4  *                      CHAR(1),       /* CHARACTERS "K"                           */
4  KEY_NUM                FIXED BIN(15), /* KEY NUMBER                               */
4  KEY_INDXLVL            FIXED BIN(15), /* #INDEX LEVEL                             */
4  KEY_NUMDATA            FIXED BIN(31), /* CI NUMBER OF DATA CHAIN BEGINNING        */
4  KEY_NUMROOT            FIXED BIN(31), /* CI NUMBER OF ROOT INDEX TREE             */
```

```
4  KEY_NUMFREE            FIXED BIN(31),  /* NUMBER OF FIRST FREE CI                */
4  KEY_NUMINDENT          FIXED BIN(15),  /* #INDEX ENTRIES/CI                      */
4  KEY_SIZE               BIT(8),         /* KEYSIZE                                */
4  KEY_LOC                FIXED BIN(15),  /* KEYLOC                                 */
4  KEY_DUPREC             BIT(8),         /* DUPREC:2(NO) 1(YES)                    */
4  *                      CHAR(2),        /* RFU                                    */
2  DATA_INFORMATION,
3  NBREC                  FIXED BIN(31),  /* NUMBER OF USER WRITTEN RECORDS         */
3  STATUS1,                               /* UFAS PHYSICAL STABILITY STATUS         */
4  OU_INST                BIT(1),         /* OUTPUT INSTABILITY                     */
4  AP_INST                BIT(1),         /* APPEND INSTABILITY                     */
4  IO_INST                BIT(1),         /* INPUT-OUTPUT INSTABILITY               */
4  UP_INST                BIT(1),         /* UPSATE INSTABILITY                     */
4  *                      BIT(4),         /* RFU                                    */
3  STATUS2,                               /* UFAS LOGICAL STABILITY STATUS          */
4  SORTIDX_OR_IDS_RCVY    BIT(1),         /* SECONDARY KEYS NOT SORTED FOR UFAS     */
                                          /* INDEXED OR IDS INSTABILITY             */
4  JOURNAL_RCVY,                          /* JOURNAL INSTABILITY                    */
5  ROLLBACK               BIT(1),         /* ABORT DURING ROLLBACK                  */
5  ROLLFORWARD            BIT(1),         /* ABORT DURING ROLLFORWARD               */
5  ABORT                  BIT(1),         /* ABORT DURING ROLLBACK OR ROLLFORWARD   */
4  *                      BIT(4),         /* RFU                                    */
3  V1_RECOVERY            BIT(1),         /* INSTABILITY ON A FILE CREATED AND      */
                                          /* PROCESSED UNDER V2 OR V3-NLS           */
3  *                      BIT(7),         /* RFU                                    */
3  *                      CHAR(13);
```

4.1.4 H_DCFMIG

Function

H_DCFMIG is used to declare the contents of an OUTFILE file record generated by the MIGOUT command when the detailed option is indicated (DTLD=1). See the *ASM7 File Migration Administrator's Guide*.

Syntax

```
$H_DCFMIG     [ PREFIX = l_identifier16]
              [,ATTRIB = l_char]
              [,{INIT | NINIT}]
              [,NLVL1 [LEVEL = i_cst]]
              ;
```

Parameters

PREFIX	Specifies a character string to be used as a prefix for the name of the structure and of each elementary item. (Default is none).
ATTRIB	Specifies the attributes of the structure. (Default is none).
INIT	Initialization of the structure. (Default is INIT except when the structure has BASED or DEFINED attributes).
NINIT	The structure initialization is not performed.
NLVL1	Prevents generation of structure level 1. (Default is: level 1 is generated).
LEVEL	Depth for structure fields, to be used with NLVL1. (Default is 2).

Expansion

The macro statement $H_DCFMIG PREFIX=L_; generates the following structure:

```
/* ----- $H_DCFMIG PREFIX=L_; ----- */
DCL 1 L_FMIG_RECORD,                          /* Migration outfile record description */
                                              /* DTLD=1 option                        */
     2 L_VERSION    CHAR  (2) INIT("01"),     /* record version (first = 01)          */
     2 L_RECTYPE    CHAR  (1) INIT("0"),      /* "0" = main record                    */
                                              /* "01" = additional volume record      */
                                              /* (multi-volume files                  */
     2 *            CHAR  (1) INIT(" "),      /* rfu                                  */
     2 L_EFN        CHAR (44) INIT((44)" "),  /* external file name                   */
     2 L_VOLMD      CHAR  (6) INIT((6)" "),   /* volume name                          */
     2 L_VOLDVC     CHAR  (8) INIT((8)" "),   /* device class name                    */
     2 L_VOLSZ      CHAR (11) INIT((11)"0"),  /* extents cumulated size on this volume*/
                                              /* (unit = KByte)                       */
     2 L_VOLSET     CHAR (6)  INIT((6)" "),   /* VOLSET name (if any)                 */
     2 L_OWNER      CHAR (12) INIT((12)" "),  /* name of project owner of the file    */
     2 L_DATES,                               /* dates (form = YYMMDD)                */
     3 L_LASTREF    CHAR (6)  INIT((6)" "),   /* last reference date                  */
     3 L_UPDATED    CHAR (6)  INIT((6)" "),   /* last update date                     */
     3 L_SAVED      CHAR (6)  INIT((6)" "),   /* save date                            */
     3 L_EXPIRD     CHAR (6)  INIT((6)" "),   /* expiration date                      */
     2 L_SIZE       CHAR (11) INIT((11)"0");  /* file size, unit = KByte (1024 bytes) */
```

4.1.5 H_DCGTSIZE

Function

Declares the structure for allocation information. This declarative provides a structure used in the output of the primitive H_GTSIZE.

Syntax

```
$H_DCGTSIZE

          [,NAME = { l_identifier31 | H_DCGTSIZE}]
          [,PREFIX = { l_identifier15| H_GTSIZE_}]
          [,ATTRIB = { l_char | AUTO}]
          [,NLVL1] ;
```

Parameters

NAME	Name of the declarative structure (H_DCGTSIZE by default).
PREFIX	Prefix of the declarative structure variables (H_GTSIZE_ by default).
ATTRIB	Used to qualify the declarative structure (AUTO by default).
NVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

```
/*__..__  $H_DCGTSIZE;                          */

DCL 1  H_DCGTSIZE AUTO,
    2  H_GTSIZE_SIZE_GLOBLOG,
    3  H_GTSIZE_GLOBAL_SIZE       FIXED BIN(31),     /* size in units of allocation       */
    3  H_GTSIZE_GLOBAL_INCRSIZE   FIXED BIN(31),     /* incrsize in units of allocation   */
    3  H_GTSIZE_GLOBAL_MAXSIZE    FIXED BIN(31),     /* maxsize in units of allocation    */
    3  *                          CHAR(8),
    3  H_GTSIZE_GLOBAL_UNIT       BIT(8),            /* unit of allocation                */
                                                     /*  volume  VBO                      */
                                                     /*    "02"X = cyl                    */
                                                     /*    "04"X = track                  */
                                                     /*  volume  FBO                      */
                                                     /*    "40"X = block                  */
    3  H_GTSIZE_LOGICAL_SIZE      FIXED BIN(31),     /* size in units of allocation       */
    3  H_GTSIZE_LOGICAL_INCRSIZE  FIXED BIN(31),     /* incrsize in units of allocation   */
    3  H_GTSIZE_LOGICAL_MAXSIZE   FIXED BIN(31),     /* maxsize in units of allocation    */
    3  *                          CHAR(8),
    3  H_GTSIZE_LOGICAL_UNIT      BIT(8);            /* unit of allocation                */
                                                     /*  files LINKQD,NONE               */
                                                     /*    "40"X = block                  */
                                                     /*  files  UFAS                      */
                                                     /*    "10"X = record                 */
```

4.1.6 H_ENDMIG

Function

Performs complementary GCOS 7 functions after a successful or an unsuccessful save server operation (migration-out or migration-in).

This primitive is intended for developing file save and file auto-retrieval servers.

For further information, refer to the manual *Introduction to ASM7*.

Syntax

```
$H_ENDMIG     efn,
              FUNCTION = i_bit8
              ,STATUS = i_fb_15
              [,OPTIONS = iv_structure]
              [,NWRJOR = i_bit1]
              [,PRTFILE = i_ptr]
              ;
```

Parameters

efn — Specifies the external file name (i_char44).

FUNCTION — Specifies the current operation as follows:
"01"X migration-out,
"02"X migration-in.

STATUS — Specifies the completion code value as indicated below (see Job Management H_SETST):

Range	SEVERITY level
0 - 99	0 (normal termination)
00 - 999 1000 - 9999	1 (normal termination) 2 (normal termination + warning)

Range	SEVERITY level
10000 - 19999	3 (abnormal termination)
20000 - 32767	4 (abnormal termination)

Example: you may specify 0 for a normal termination and 10000 for an abnormal one.

OPTIONS — Specifies the name of an input structure declared by the H_DCENDMIG primitive.
This structure contains Save Server specific parameters.

NWRJOR — Indicates whether or not H_ENDMIG may write completion messages on the Job Occurrence Report (JOR).

"1"B completion messages are not written on JOR,
"0"B all messages are written on JOR (default).

PRTFILE — Specifies the FD address of a report file.
When specified, the JOR messages are (also) written onto this file.

Normal return codes

DONE: — H_ENDMIG successfully completed.

Abnormal return codes

ARGERR: — Wrong value for service name.

FILEMIG: — Abnormal migration-in termination (STATUS>9999).

FUNCNAV: — ARS not available on this site.

NOTDONE: — Abnormal migration-out termination (STATUS>9999).

PARAMERR: — Erroneous parameter.

Abnormal Return Codes issued from: H_RTFLCAT, H_DEALLOC, H_MODIFY, e.g.:

OBJUNKN	The cataloged file name referenced by efn does not exist.
ARVIOL	Access right violation.

Comments

This primitive is called by a Save Server to complete a migration-out or a migration-in operation, after saving or restoring a file. The file must have been deassigned before calling H_ENDMIG. GCOS 7 complementary functions are described below.

Migration-out:

Migration-out (normal termination):

The file disk space is released. The file catalog entry is updated and set to “migrated” state.

If NWRJOR = “0”B, the following message is written on the JOR:

HM40 SUCCESSFUL FILE MIGRATION-OUT: <efn>

Migration-out (abnormal termination):

If NWRJOR = “0”B, the following message is written on the JOR:

HM42 UNSUCCESSFUL FILE MIGRATION-OUT: <efn>

RC=xxxxxxxx->RC-char30, ERROR-CODE= <error-number>

where: ERROR-CODE is equal to the STATUS value.

Migration-in:

Migration-in (normal termination):

The migrated information is reset in the cataloged file description. The file becomes available for any user.

Any process group waiting for the file restoration (automatic migration-in operating mode applied) is reactivated.

If NWRJOR = "0"B, the following message is written on the JOR:
HM41 SUCCESSFUL FILE MIGRATION-IN: <efn>

Migration-in (abnormal termination):

If NWRJOR = "0"B, the following message is written on the JOR:
HM43 UNSUCCESSFUL FILE MIGRATION-IN: <efn>
RC=xxxxxxxx->RC-char30, ERROR-CODE=<error-number>
where: ERROR-CODE is equal to the STATUS value.

The migrated information is kept in the cataloged file description. The file remains in "migrated" state.

The H_ENDMIG <migration-in> primitive cannot be called for "pre-migrated" files. The following message is written on the JOR:
HM45 ABNORMAL USE OF H_ENDMIG MIGIN ON PRE-MIGRATED FILE: <efn>

If CLEAN is set to "1"B in migration-in options (H_DCENDMIG) the file disk space is released,

If NLAST is set to "0"B in migration-in options (H_ENDMIG) or OPTIONS is not specified: any process group waiting for the file restoration (automatic migration-in operating mode applied) is reactivated.

4.1.7 H_FILINFO

Function

Gets information from the UFAS label of an unopened UFAS file.

Syntax

```
$H_FILINFO
          IFNPTR
          ,WAPTR ;
```

Parameters

IFNPTR	Pointer on the internal file name.
WAPTR	Pointer on the user-defined working area which is to receive data read on the UFAS label. The structure is defined by the declarative H_DCFILINFO.

Normal return codes

DONE:	H_FILINFO successfully completed

Abnormal return codes

ARGERR:	Argument error: erroneous or missing parameters.
CATERR:	Assignment anomaly: contradictory file status (cataloged or uncataloged) for the UFAS file referenced by the IFNPTR parameter.
EFNUNKN:	FileUFAS file referenced by the IFNPTR parameter exists but is:

– either unformatted (the UFAS extended label is missing)
– or physically in an unstable state (i.e., partially restored/duplicated by Data Management utilities)

MDNAV:	Media not available.
NOTDONE:	Environment of the execution is forbidden (call from TDS).
OPEN:	File already opened.
WAOV:	Working area overflow, i.e. user buffer overflow.

Comments

- H_FILINFO can be used to read UFAS label for all UFAS organizations i.e. SEQ, REL, IND or IDS.
- Several functions H_FILINFO can be activated in the same step, but H_FILINFO cannot be used in a step which already uses the H_RFLDEF or DEFINE primitives.
- The normal use of H_FILINFO is on a closed file which is not being concurrently used by other steps (but this condition is not checked). H_FILINFO opens the file, reads the label, and then closes the file.
- Share rules are those defined by the assignment statement and can be:
  – either ACCESS = READ SHARE = MONITOR
  – or ACCESS = WRITE SHARE = MONITOR
- Use of H_FILINFO is forbidden in the TDS environment.
- For UFAS SEQ, IND, or REL files, H_FILINFO returns the number of records currently contained by the file and places this value in the DATA_INFORMATION.NBREC field. This information is valid for files allocated from TS 7356 (or later) and for files allocated earlier than TS 7356 but which have been loaded (by LOAD_FILE) in output mode in TS 7356 (or later). If the information is not valid, the NBREC field is set to -1. At file close time, the UFAS Access Method writes the NBREC information in the file label. Therefore, if H_FILINFO does not have exclusive access to

the file and it is shared concurrently by one or more steps, the NBREC information may not be accurate.

4.1.8 H_GTSIZE

Function

Returns the size and increment for a file or a library and the maximum size for a library. The returned values are in logical and physical allocation units.

Syntax

```
$H_GTSIZE
         ifn
         ,OUTSTRUCT = o_structure ;
```

Parameters

ifn	i_char8 Specifies the internal file name of the file to be processed.
OUTSTRUCT	Name of the structure declared by H_DCGTSIZE. This structure contains the desired results only if the return code is DONE.

Normal return codes

DONE:	The o_structure contains the results.

Abnormal return codes

WRONGORG:	the file is UFAS EXTENDED on a system not UFAS LS configured.
EFNUNKN:	the file does not exist.
IGNORE:	the ifn was not specified or none of the file supporting volumes where mounted.

DATANAV:	the file is unstable.

Other return codes may be returned by the system.

Comments

The H_GTSIZE primitive fills the structure referenced by the OUTSTRUCT parameter.

This structure is made up of two parts:

- a physical part which contains the size, increment size and maxsize in physical allocation units (BLOCK, CYL or TRACK),
- a logical part which contains the size, increment size and maxsize in logical allocation units (BLOCK for LINKQD and NONE files, RECORD for the others).

The primitive must be called before opening the file and after a call to the H_RTFLDEF primitive (in order to sort the volumes by ascending VSEQNs). The volumes (only disks) supporting the file must be mounted and accessible.

The types of file organization supported are as follows:

FBO volume:	LINKQD UFAS LS IDS2 NONE
VBO volume:	LINKQD UFAS LS UFAS NLS IDS2 BFAS SEQ NONE

4.1.9 H_IFNASG

Function

Lists the ifns that are assigned to a step by means of ASSIGN JCL statement or H_ASSIGN dynamic primitive.

Syntax

```
$H_IFNASG
         ifn-list
         ,NBENTRY = i_fb15 ;
```

Parameters

ifn-list	o_array Array which is reserved by the caller to be filled with the ifn's. `DCL ARRAY (NBENTRY)  CHAR (8);`
NBENTRY	Number of entries of the array reserved by the caller.

Normal return codes

DIRLIM:	All the IFN's have been stored within the given array.

Abnormal return codes

WAOV:	Array overflow.

Comments

The entries of the array which are not filled are set to blanks.

4.1.10 H_LOAD_FILE

Function

Loads or reorganizes a UFAS file. Also loads a library member.

Syntax

```
$H_LOAD_FILE
          infile
          ,outfile
          [,START = i_fb31]
          [,INCR = i_fb31]
          [,HALT = i_fb31]
          [,PMD = i_char2]
          [,NORDER = i_bit1]
          [,OUTALC = i_bit1]
          [,CATNOW = i_bit1]
          [,FMEDIA = i_bit1]
          [,NWRJOR = i_bit1]
          [,DSLFILE = i_ptr] ;
```

Parameters

infile	specifies the internal file name (declared by H_FD) of the input file that contains the data to be loaded.
outfile	specifies the internal file name (declared by H_FD) of the output file to be loaded.
START	specifies the position, in the input file, of the first record to be loaded. Default value is 1.

INCR	specifies the increment, in the input file, between successive records to be loaded. Default value is 1.
HALT	specifies the number of records to be loaded from the input file. Default value is the last input file record.
PMD	specifies the processing mode of the output file to be loaded. "AP": the processing mode is APpend. The input file contents is appended at the end of the existing data of the output file. "OU": (default) the processing mode is OUtput. The output file is loaded starting from its beginning. Any existing data is overwritten.
NORDER	specifies, for an output UFAS INDEXED file, whether or not the loading is to be done in the correct key order. "1"B: the data records of the input file are not in primary key order. "0"B: (default) the data records of the input file are not in ascending order of primary key values.
OUTALC	indicates whether or not the missing output file is to be dynamically allocated taking the input file as model. "1"B: dynamically allocates missing output file before the loading takes place. "0"B: (default) uses the existing output file.
CATNOW	indicates whether or not the output file is to be cataloged during dynamic allocation. "1"B: the output file is to be cataloged during dynamic allocation. OUTALC must be set to "1"B,

	and FILESTAT="04"X (UNCAT) must be specified for the output file in H_ASSIGN. "0"B: (default) the output file is not to be cataloged during dynamic allocation.
FMEDIA	for a cataloged output file, specifies whether or not the existing cataloged list of volumes is to be erased and replaced by the volumes (given in H_ASSIGN) onto which dynamic allocation took place. "1"B: replace in the catalog, for the output file, the existing list of volumes. OUTALC="1"B and CATNOW="1"B must also be specified. "0"B: (default) do not replace in the catalog, for the output file, the list of volumes: the file level catalog entry is supposed to be empty.
NWRJOR	indicates whether or not H_LOAD_FILE may write messages on the Job Occurrence Report (JOR). "1"B: no message is written on the JOR. "0"B: (default) messages are written on the JOR.
DSLFILE	specifies the address of the internal file name (declared by H_FD) of the sequential input file that contains the DSL (Data Service Language) commands to be executed. The DSL is fully described in the *Data Management Utilities User's Guide*. When this parameter is not specified or is given a NULL() value, no DSL file is to be used (default value).

Normal return codes

DONE:	H_LOAD_FILE is successfully done. Outfile is loaded.

Abnormal return codes

ARGERR:	one or more given parameters are wrong.
ARVIOL:	access rights violation. The user does not have the required access rights on either the input or the output file or the DSL file.
CATERR:	assignment of either the input file or the output file or the DSL file contradicts their actual cataloged or uncataloged file status.
DATANAV:	either the input file or the DSL file is in an unstable state.
DVNAV:	there are not enough devices (disk or tape drives) currently available to meet the request. The DEFER option was used for a file assignment.
DUPKEY:	one (or more) record to be loaded has a duplicated key field.
EFNUNKN:	either the input file or the output file or the DSL file does not exist on the list of volumes retrieved from H_ASSIGN. OUTALC="1"B may be used for a missing output file.
EXHAUST:	the end of outfile has been reached and either no increment size has been specified at allocation time or there is no more free space on the media list that support outfile.
IFNERR:	either the infile or the outfile or the DSLFILE parameter is wrong, leading to invalid control structures.
IFNNASG:	one of the files, input or output or DSL is not assigned. At least, both input and output files should be assigned.

IOFAIL:	an irrecoverable read error was encountered either on the input file or on the DSL file.
MDNAV:	a volume mounting request has been canceled by the operator.
MDERR:	incorrect or partial media list given for infile, outfile or DSL file.
NOTOBS:	the outfile tape to be overwritten is still valid, the expiry date has not been reached.
NOWAIT:	synchronization error. Infile is being accessed by another task.
OPEN:	at least one of the files, infile, outfile or DSL file is open. All these files should be closed before H_LOAD_FILE is issued.
OPTERR:	one of the files, infile, outfile or DSL file is a UFAS file that has been assigned with SHARE=MONITOR and the REPEAT parameter has not been specified in the STEP using H_LOAD_FILE.
OUTSEQ:	either input records are not given in a correct key order or NORDER="1"B has not been specified.
TRUNC:	outfile record length is shorter that infile record length. Outfile is loaded but some records may have been truncated.
SPACENAV:	there is not enough available space on the list of volumes specified in H_ASSIGN to allocate the output file.
YETALLOC:	the catalog entry referenced by the output file already contains a list of volumes. FMEDIA="1"B must be used to continue.

Some other return codes may also be returned to the caller especially when OUTALC="1"B is specified to request dynamic allocation of the missing output file (refer to H_PREALLOC in the *GPL System Primitives Reference Manual*).

Comments

- H_LOAD_FILE copies the records of the input file (as referenced by the infile parameter) into the output file (as referenced by the outfile parameter).
- Both infile and outfile must be assigned and must be closed when H_LOAD_FILE is called. Furthermore, if DSL commands are to be used, the DSL file must also be assigned and closed.
- To get a cataloged outfile, both CATNOW="1"B and OUTALC="1"B must be specified and FILESTAT="04"X (UNCAT) must be specified in H_ASSIGN. When the H_LOAD_FILE operation is over, the outfile's status changes from "04"X (UNCAT) to "00"X (CAT).
- Supported files are: UFAS SEQUENTIAL (disk and tape), UFAS RELATIVE and UFAS INDEXED. LIBRARY MEMBERS (or SUBFILES) are also supported.
- When at least one file is a library member, the following options must be used at the linking time of the load-module :
  - USEDSM=H_SM9,
  - STARTASG=(PRIVATE=11),
- The files may have the record format fixed or variable, blocked or unblocked.
- When input records are larger than the output records, the input records are truncated on the right to conform to the maximum output record length. In this case, H_LOAD_FILE returns to the caller the TRUNC return code but outfile is fully loaded.

  When input records are shorter than the output records, and the output records have the format fixed or fixed unblocked, the input records are expanded to match the output records. In this case, the space which is added to the input records is filled with space characters.
- H_LOAD_FILE accesses the input file in a sequential manner. If the output file is a UFAS INDEXED file, records should preferably be delivered in the ordered sequence based upon the key field (using NORDER="0"B) but, such a file may also be loaded with records supplied in unordered sequence (using NORDER="1"B).

- When a UFAS RELATIVE file is copied into a UFAS relative file, the access mode is forced to direct in order to preserve any deleted records in the output file.
- A UFAS INDEXED file with secondary keys may also be loaded by using H_LOAD_FILE but in this case, the SORT_INDEX utility must be executed before any secondary index access.
- It is possible, using H_LOAD_FILE, to load a library member (or subfile). In this case, the loaded member has no control record and is given a SARF type. Conversion from SARF to SSF may be done by the MAINTAIN_LIBRARY MOVE command.
- If PMD="AP" is specified then, the input records are added sequentially after the last record currently present in the output file.
- The Data Service Language (DSL) is available through the DSLFILE parameter. DSL commands, from DSL file, allows to select only certain records from the input file and load them into the output file.
- In addition to the record selection performed by the DSL, the user can ask H_LOAD_FILE to ignore n-1 records of the input file and start processing at the nth input record; this is done by using the START parameter. Similarly, the user can ask H_LOAD_FILE to stop processing the input file as soon as p records have been written to the output file; this is done by using the HALT parameter. It is also possible to select one input record every m records; this is done by using the INCR parameter.
- If the output file has been given an increment size at creation time and the end of allocated space is encountered while loading then, it is automatically incremented by the system.
- Whenever a malfunction occurs during the execution of an H_LOAD_FILE operation, the malfunction is reported to the user via JOR messages (except when NWRJOR="1"B is specified) and via return codes. The step completion code is then forced to SEV1 regardless of its previous value.

For more information, refer to the CREATE utility in the *Data Management Utilities User's Guide*.

4.1.11 H_MODIFY

Function

To modify a cataloged file (or generation) description.

Syntax

```
$H_MODIFY
          efn
            { NMEDIA = i_bit1                          }
          [,{                                          }]
            { MEDIA = i_structure[,EXTMDLST=i_bit1]    }
          [,CATALOG = i_bit8]
          [,SHARE = i_bit8]
          [,DUALSHR = i_bit8]
          [,JOURNAL = i_bit8]
          [,ALOCK = i_bit1]
          [,SLOCK = i_char3]
          [,INCRSIZE = i_fb31]
          [,UNIT = i_bit8]
          [,IOC = i_char1]
          [,VOLSET = i_char6]
          [,LOGSUBF = i_char1]
          [,MIGRATION = i_logbin8]
          [,RESIDENCY = i_logbin8]
          [,NPROTECT];
```

Parameters

efn	i_char44 mandatory parameter that specifies the external file name of the file whose catalog description is to be modified.
NMEDIA	erases the media list from the referenced file's catalog description. NMEDIA and MEDIA are mutually exclusive. "1"B: erase the current media list "0"B: (default) no action.
MEDIA	name of the structure (declared by $H_DCMEDIA) that contains the list of volumes used to update or extend the current media list. MEDIA and NMEDIA are mutually exclusive.
EXTMDLST	extends the referenced file's current media list to include volume(s) specified via MEDIA. These additional volumes will be used when necessary in the case of dynamic extension. "1"B: extend the current media list. "0"B: (default) no action.
CATALOG	rank of the CATALOG (within the attached catalogs) in which the referenced file is to be searched. "01"X to "05"X: only search within the specified catalog. "00"X: (default) apply catalog search rules. "FF"X: only search within the SITE.CATALOG.
SHARE	new level of concurrent access to the referenced file. "80"X: NORMAL - several readers or one writer. Sharing is controlled at file level. "10"X: DIR - several readers or one writer. Sharing is controlled at subfile level. Strongly recommended for libraries.

"40"X: ONEWRITE - several readers and one writer. Sharing is controlled at file level.

"20"X: MONITOR - several readers and writers. Under General Access Control (GAC), sharing is controlled at UFAS Control Interval level.

"08"X: FREE - unlimited concurrent access is authorized. File consistency is not guaranteed (except with catalogs) and is the user's responsibility.

"04"X: UNSPEC - no sharing condition in the catalog. Sharing conditions are taken from the ASSIGN, ASGi or H_ASSIGN statements.

"00"X: (default) no action.

DUALSHR — new file sharing level between dual systems for the referenced file

"00"X: NONE - the file can be accessed by only one system at a time.

"80"X: NORMAL - the file can be read by both systems or written by only one.

"40"X: ONEWRITE - one system may only read when one writes (this can result in "dirty reads").

"08"X: FREE - unlimited access by both systems. The user is responsible for ensuring file consistency.

JOURNAL — new journalization options for the referenced file or file generation of a generation group.

"00"X: NO - no journal is used.
"01"X: BEFORE - before journal is to be used.
"02"X: AFTER - after journal is to be used.
"03"X: BOTH - both before and after journals are to be used.

"04"X: PRIVATE - user-developed journalization routines are to be used.

ALOCK	new abort lock status for the referenced file. The abort lock mechanism is activated when the file has the PRIVATE journalization option. If a step or a system check occurs after the file or generation has been opened in a state other than INPUT, the system sets the abort lock on the file. No user can access the file except by specifying ACCESS = RECOVERY in an ASSIGN statement, for salvaging purposes. The abort lock is reset if the user salvaging step terminates normally. "1"B: the file is set in the abort lock state. "0"B: the file is reset in the non-abort lock state.
SLOCK	new type of security lock for the referenced file. "OFF": no security lock (it is turned off). "IN": file can be processed in INPUT mode only. "AP": file can be processed in APPEND mode only. "IA": file can be processed in INPUT or APPEND mode "IO ": file can be processed in INPUT, OUTPUT or APPEND mode. " ": (default) no action.
INCRSIZE	new increment size for the referenced file. This is the amount of storage space to be added automatically whenever the file's allocated space is completely used. Note that H_MODIFY changes only the value of INCRSIZE in the catalog; it does not change the value in the file label. For a generation group, the INCRSIZE parameter applies to each generation of the generation group. If you want a different INCRSIZE for each generation, you should use INCRSIZE at each generation, and not at the generation group level. Possible value is from 0 to 32767.

UNIT	unit of allocation for when INCRSIZE is specified. "04"X: TRACK "02"X: CYLINDER "40"X: BLOCK
IOC	new I/O cache mode for the referenced file. "B": bypass the use of I/O cache. "F": force the use of I/O cache. "D": use the default I/O cache mode (as defined by the START_IO_CACHE command). " ": (default) no action. Note that the use of the I/O cache may also be controlled by the BPIOC and FRIOC DEFINE parameters. These parameters, if specified, override the IOC cataloged value.
VOLSET	sets or clears the Volset entry for the referenced file. value: the name of the Volset on which the file is allocated. blanks: erase the Volset name currently registered in the file's catalog description. "000000000000"H: (default) no action.
LOGSUBF	specifies whether or not modifications to subfiles of the referenced library are to be logged. "Y": log subfiles events. "N": do not log subfiles events. " ": (default) no action.
MIGRATION	Sets or clears the pre-migration or migration information for the referenced file. This parameter should only be used for cataloged disk files which are concerned by migration to secondary storage using an archive management tool. – the file has been migrated using a user-developed tool. – the file has been pre-migrated using a user-developed tool.

– erase the pre-migration or migration information. The file is no longer considered to be migrated.

CAUTION:
When migration information is set, the file's UNIT, SIZE, INCRSIZE and media list are automatically deleted from the file's catalog description.

RESIDENCY — Affects the residency factor for the referenced file. This parameter should only be used for cataloged disk files which are concerned by migration to secondary space using an archive management tool.
The residency factor is used by the migration mechanism to select files, space of which is to be deleted first.

0 : (default) no action:
1 to 15 : the file has the corresponding residency factor.
255 : erase the residency factor information:

NPROTECT — specifies that the referenced tape file is no longer protected. There will be no cataloged flag in the file label at the next open output.

Normal return codes

DONE: — H_MODIFY is successfully done.

Abnormal return codes

ARGERR:	One or more erroneous parameters. (For example, MEDIA and NMEDIA given simultaneously, more than 10 volumes specified by MEDIA, wrong value for the SHARE parameter).
ARVIOL:	Access rights violation. The user does not have the OWNER access right on the referenced file and so is not allowed to modify its catalog description.
IOFAIL:	Unrecoverable I/O error on the catalog.
OBJUNKN:	The referenced file does not exist or cannot be found in the catalog specified.
OPTERR:	Illegal value specified for the MIGRATION parameter.
SCATUNKN:	The site catalog does not exist.
SCATVIOL:	The given catalog rank does not lead to a valid catalog (DUMMY catalog or catalog rank does not exist in ATTACH statement).

Comments:

- You can use the H_MODIFY primitive to ensure that a file's cataloged residency always corresponds exactly with its actual residency.
- H_MODIFY modifies the list of volumes which support a file without control (it is merely a catalog operation and does not require volume mounting). When extending the media list (EXTMDLST parameter), you must ensure that the additional volumes are of the same type as the current volumes.
- H_MODIFY requires the OWNER access right on the file concerned.

For more information concerning cataloged files and their attributes, refer to the *Catalog Management User's Guide*.

For more information concerning the use of the MIGRATION parameter, refer to the *ASM 7 File Migration Administrator's Guide*.

4.1.12 H_PRINT_FILE

Function

To print records from a file or a library member.

Syntax

```
$H_PRINT_FILE
          infile
          [,START = i_fb31]
          [,INCR = i_fb31]
          [,HALT = i_fb31]
          [,FORMAT = i_char5]
          [,DSLFILE = i_ptr];
```

Parameters

infile	specifies the internal file name (declared by H_FD) of the input file that contains records to be printed.
START	specifies the position, in the input file, of the first record to be printed. Default value is 1.
INCR	specifies the increment, in the input file, between successive records to be printed. Default value is 1.
HALT	specifies the number of records to be printed from the input file. Default value is the last input file record.
FORMAT	specifies the format in which the records are to be printed. "ALPHA": records are printed in alphabetic characters (default value).

	"HEXA": records are printed in hexadecimal format.
	"BOTH": records are printed in both alphabetical and hexadecimal formats.
DSLFILE	specifies the address of the internal file name (declared by H_FD) of the sequential input file that contains the DSL (Data Service Language) commands to be executed. The DSL is fully described in the *Data Management Utilities User's Guide*. When this parameter is not specified or is given a NULL() value, no DSL file is to be used (default value).

Normal return codes

DONE:	H_PRINT_FILE is successfully done. Infile is printed.

Abnormal return codes

ARGERR:	one or more given parameters are wrong.
ARVIOL:	access rights violation. The user does not have the required access rights on either the input file or the DSL file.
CATERR:	assignment of either the input file or the DSL file contradicts their actual cataloged or uncataloged file status.
DATANAV:	either the input file or the DSL file is in an unstable state.
DVNAV:	there are not enough devices (disk or tape drives) currently available to meet the request. The DEFER option was used for a file assignment.
EFNUNKN:	either the input file or the DSL file does not exist on the list of volumes retrieved from H_ASSIGN.

IFNERR:	either the infile or the DSLFILE parameter is wrong; that leads to invalid control structures.
IFNNASG:	one of the files, input or DSL is not assigned. At least infile should be assigned.
IOFAIL:	an irrecoverable read error was encountered either on the input file or on the DSL file.
MDNAV:	a volume mounting request has been canceled by the operator.
MDERR:	incorrect or partial media list given for infile or DSL file.
NOWAIT:	synchronization error. Either infile or DSL file is being accessed by another task.
OPEN:	at least one of the files, infile or DSL is open. All these files should be closed before H_PRINT_FILE is issued.
OPTERR:	one of the files, infile or DSL file is a UFAS file that has been assigned with SHARE=MONITOR and the REPEAT parameter has not been specified in the STEP using H_PRINT_FILE.

Comments

- H_PRINT_FILE prints logical records from the input file (as referenced by the infile parameter).
- Infile must be assigned and closed when H_PRINT_FILE is called. Furthermore, if DSL commands are to be used, the DSL file must also be assigned and closed.
- Supported files are: UFAS SEQUENTIAL (disk or tape), UFAS RELATIVE and UFAS INDEXED. LIBRARY MEMBERS (or SUBFILES) are also supported.
- H_PRINT_FILE accesses the input file in a sequential manner so, records are printed sequentially.

- The Data Service Language (DSL) is available through the DSLFILE parameter. DSL commands, from DSL file, allows to select only certain records from the input file and load them into the output file.
- In addition to the record selection performed by the DSL, the user can ask H_PRINT_FILE to ignore n-1 records of the input file and start printing at the n'th input record; this is done by using the START parameter. Similarly, the user can ask H_PRINT_FILE to stop printing the input file as soon as p records have been printed; this is done by using the HALT parameter. It is also possible to select one input record every m records; this is done by using the INCR parameter.
- In IOF mode, the output file is the terminal and needs not to be specified. In BATCH mode, there is no default for the output file so it must be specified either statically in the basic JCL of the launching STEP or dynamically via H_ASSIGN. More, the internal file name (ifn) to be used in assignment must be H_PRTN, ex:

  – ASSIGN H_PRTN, SYS.OUT,
  or
  – ASSIGN H_PRTN, MY.REPLIB, MB=MYREP;

- Whenever a malfunction occurs during the execution of a H_PRINT_FILE operation, the malfunction is reported to the user via JOR messages (except when NWRJOR="1"B is specified) and via return codes. The step completion code is then forced to SEV1 regardless of its previous value.

For more information, refer to the PRINT utility in the *Data Management Utilities User's Guide*.

4.1.13 H_RESTORE_FILE

Function

Restores the contents of a disk file from a sequential UFAS disk file or tape file.

Syntax

```
$H_RESTORE_FILE
          infile
          ,outfile
          [,SAVENAME = i_char44]
          [,NCKDATE = i_bit1]
          [,OUTALC = i_bit1]
          [,CATNOW = i_bit1]
          [,FMEDIA = i_bit1]
          [,MIGRATION = i_bit1]
          [,NWRJOR = i_bit1] ;
```

Parameters

infile	specifies the internal file name (declared by H_FD) of the input file that contains the image of the file to be restored. This file must be a UFAS disk file or tape file created either by a previous GPL primitive (H_SAVE_FILE, H_SAVE_DISK, H_FILSAVE) or by a previous JCL/GCL command (FILSAVE, VOLSAVE, SAVE_FILE(SET), SAVE_CATALOG, SAVE_DISK).
outfile	specifies the internal file name (declared by H_FD) of the output file to be restored.

NCKDATE	indicates whether or not the date of the save image to be restored from the input file is to be checked. "1"B: do not check the save image date before restoring. Note that such a restore may induce some errors. "0"B (default): check the save image date before restoring.
SAVENAME	specifies the name of the saved file image to be restored. By default this parameter is given the output file name.
OUTALC	indicates whether or not the missing output file is to be dynamically allocated taking the input file as model. "1"B: dynamically allocates missing output file before the restore takes place. Note that if an output file with the same characteristics already exists when H_RESTORE_FILE is issued and this file is too small, then this file will be extended in order to contain the whole file to be restored. "0"B: uses the existing output file (default value).
CATNOW	indicates whether or not the output file is to be cataloged during dynamic allocation. "1"B: the output file is to be cataloged during dynamic allocation. OUTALC must be set to "1"B, and FILESTAT="04"X (UNCAT) must be specified for the output file in H_ASSIGN. "0"B (default): the output file is not to be cataloged during dynamic allocation.
FMEDIA	for a cataloged output file, specifies whether or not the existing cataloged list of volumes is to be erased and replaced by the volumes (given in H_ASSIGN) onto which dynamic allocation took place.

"1"B: replace in the catalog, for the output file, the existing list of volumes. OUTALC="1"B and CATNOW="1"B must also be specified.
"0"B (default): do not replace in the catalog, for the output file, the list of volumes. The file level catalog entry must be empty.

MIGRATION — for a missing cataloged output file, specifies whether or not the restore is relative to a migration.

"1"B: the restore is relative to a migration operation.
"0"B: the restore is not relative to a migration operation.
If MIGRATION="1"B, CATNOW and OUTALC must both also be set to "1"B otherwise the return code ARGERR is produced.

NWRJOR — indicates whether or not H_RESTORE_FILE may write messages on the Job Occurrence Report (JOR).

"1"B: no message is written on the JOR.
"0"B: messages are written on the JOR (default value).

Normal return codes

DONE: H_RESTORE_FILE is successful. Outfile is restored.

DONEID: H_RESTORE_FILE is successfully completed but some parts of outfile has not been restored. This is either because the H_SAVE_FILE which saved the file terminated with the return code DONEID, or because the ERROPT parameter was used in H_FD for outfile. The integrity and consistency of the restored file is not assured.

Abnormal return codes

ARGERR:	one or more parameters are wrong (incompatible device class, input file is neither a UFAS sequential disk file nor a tape file, infile is a DUMMY file, logical characteristics of the file to be restored do not match the outfile logical characteristics. MIGRATION is set to "1"B when OUTALC and/or CATNOW is set to "0"B, etc...).
ARVIOL:	access rights violation. The user does not have the required access rights on either the input or the output file.
CATERR:	assignment of either the input file or the output file contradicts their actual cataloged or uncataloged file status.
CONFLICT:	infile has an incorrect H_FD or was not created by a save GPL primitive or JCL/GCL command.
DATALOSS:	save file was incomplete, restore is not performed.
DATANAV:	the input file is in an unstable state.
DVNAV:	there are not enough devices (disk or tape drives) currently available to meet the request. The DEFER option was used for a file assignment.
EFNUNKN:	either the input file or the output file does not exist on the list of volumes retrieved from H_ASSIGN. OUTALC="1"B may be used for a missing output file.
EXHAUST:	the end of outfile has been reached and OUTALC="1"B has not been specified for extension.
FILEMIG:	the MIGRATION parameter is set to "0"B and the catalog entry for outfile has been found migrated.
IFNERR:	either the infile or the outfile parameter is wrong; that leads to invalid control structures.

IFNNASG:	at least one of the files, input or output is not assigned. Both files should be assigned.
IOFAIL:	an irrecoverable read error was encountered either on the input file or on the output file.
MDNAV:	a volume mounting request has been canceled by the operator.
MDERR:	incorrect or partial media list given for infile or outfile.
NOMATCH:	the named component (retrieved either from the SAVENAME parameter or from the H_ASSIGN) has not been found within infile. Note that NOMATCH belongs to the NAMEERR subclass of return codes.
NOWAIT:	synchronization error. Infile is being accessed by another task.
OPEN:	at least one of the files, infile or outfile is open. All these files should be closed before H_RESTORE_FILE is issued.
TRUNC:	refer to the EXHAUST return code.
SPACENAV:	there is not enough available space on the list of volumes specified in H_ASSIGN to allocate the output file.
YETALLOC:	the catalog entry referenced by the output file already contains a list of volumes. FMEDIA="1"B must be used to continue.

Some other return codes may also be returned to the caller especially when OUTALC="1"B is specified to request dynamic allocation of the missing output file (refer to H_PREALLOC in the *GPL System Primitives Reference Manual*).

Comments

- H_RESTORE_FILE restores the contents of a disk file from a sequential UFAS disk file or tape file previously saved by a save operation.
- To get a cataloged outfile, both CATNOW="1"B and OUTALC="1"B must be specified and FILESTAT="04"X (UNCAT) must be specified in H_ASSIGN. When the H_RESTORE_FILE operation is over, the outfile's status changes from "04"X (UNCAT) to "00"X (CAT). Also, if outfile is a catalog, it is deassigned.
- In order to reserve resources needed by H_RESTORE_FILE for execution, the following option must be used at linking time of the load module: VACSEG=(SHARE=+10). In addition, one type-2 vacant entry in STN 1, 2 or 3 must be reserved, for example: SEGTAB3=(SHRLEVEL=2,VSEG=1).
- If MIGRATION = "1"B is specified for automatic retrieval of the missing outfile, the file must have been previously migrated using a specific migration tool. See the ASM 7 documentation referred to below.
- Whenever a malfunction occurs during the execution of an H_RESTORE_FILE operation, the malfunction is reported to the user via JOR messages (except when NWRJOR="1"B is specified) and via return codes. The step completion code is then forced to SEV1 regardless of its previous value.

For more information, refer to the following documents:

- *GPL System Primitives Reference Manual*, H_FILREST primitive
- *Data Management Utilities User's Guide*, FILREST utility
- *ASM 7 File Migration Administrator's Guide*.

4.1.14 H_SAVE_DISK

Function

Saves the contents of a disk volume into a sequential UFAS disk file or tape file.

Syntax

```
$H_SAVE_DISK
          iv_invol
          ,outfile
          [,VTOC = i_bit1]
          [,DIRTY = i_bit1]
          [,NWRJOR = i_bit1];
```

Parameters

iv_invol	specifies the name of the structure (declared by H_DCMEDIA) that contains the description of the disk volume to be saved. Note that only one disk volume can be saved by H_SAVE_DISK call.
outfile	specifies the internal file name (declared by H_FD) of the output file that is to be used as SAVE file. The output file must be either a UFAS sequential disk file or a tape file.
VTOC	indicates whether only the VTOC is to be saved or the entire disk volume. "1"B: save only the VTOC of the disk volume. "0"B: save the entire disk volume (default value).

DIRTY	indicates whether or not non-exclusive disk volume access is required. "1"B: the disk volume to be saved may be concurrently accessed in read or write mode by other jobs. "0"B: exclusive access is required to the disk volume (default value).
NWRJOR	indicates whether or not H_SAVE_DISK may write messages to the Job Occurrence Report (JOR). "1"B: no message is written to the JOR. "0"B: messages are written to the JOR (default value).

Normal return codes

DONE:	H_SAVE_DISK is successfully done. INVOL has been saved.

Abnormal return codes

ARGERR:	one or more given parameters are wrong.
ARVIOL:	access rights violation. The user does not have the required access rights on either the input disk volume or the output file.
CATERR:	assignment of the output file contradicts its actual cataloged or uncataloged file status.
CONFLICT:	infile has an incorrect H_FD or was not created by a save GPL primitive or JCL/GCL command.
DATANAV:	the input disk is in an unstable state.
DVNAV:	there are not enough devices (disk or tape drives) currently available to meet the request. The DEFER option was used for outfile assignment.
EFNUNKN:	outfile has not been found on the specified volumes.

EXHAUST:	the end of outfile has been reached or there is no more free space on the media list that support outfile.
IFNERR:	either the INVOL parameter or the outfile parameter is wrong; that leads to invalid control structures.
IFNNASG:	the output file is not assigned.
IOFAIL:	an irrecoverable read error was encountered on the input disk volume.
MDNAV:	a volume mounting request has been canceled by the operator.
MDERR:	incorrect or partial media list given for outfile.
NOTOBS:	the outfile tape to be overwritten is still valid, the expiry date has not been reached.
OPEN:	outfile is open. It should be closed before H_SAVE_DISK is issued.
TRUNC:	refer to the EXHAUST return code.

Comments

- H_SAVE_DISK saves the contents of a disk volume (as referenced by the INVOL parameter) into a UFAS sequential disk file or tape file (as referenced by the outfile parameter).
- Outfile must be assigned and closed when H_SAVE_DISK is called. Note that if outfile is an existing UFAS sequential disk file, it must have been allocated with RECSIZE=6132 and RECFORM=V.
- The entire contents of the disk volume may be saved or only the VTOC (Volume Table Of Contents). When the whole disk volume is saved, the VTOC is saved first as a file, and then all the files described in the VTOC.
- The file format used to save the disk volume is the same as the one created by the VOLSAVE JCL utility or the SAVE_DISK GCL command so, it may used as input file in a further VOLREST or RESTORE_DISK operation. This file may also be used as an input file by the following JCL/GCL

commands: FILREST, RESTORE_FILE, RESTORE_FILESET or the following GPL primitives: H_FILREST, H_RESTORE_FILE but in this case, only monovolume files can be restored.

- By default (or when DIRTY="0"B), H_SAVE_DISK requires exclusive access to the disk volume to be saved. If this condition is not met, the primitive does not wait until the volume is made free and the BUSY return code is returned. When DIRTY="1"B is specified, concurrent accesses by other jobs, in read or write mode, are authorized and the resulting save may be considered as "dirty". Note that if the disk volume to be saved is a RESIDENT and/or SHARED volume, the DIRTY parameter is forced to "1"B.
- Because of the duration of a save disk operation, the normal use of H_SAVE_DISK is in batch mode. This primitive may also be used in interactive mode but in this case, the terminal is made busy during all the save time. H_SAVE_DISK must not be used under TDS.
- In order to reserve resources needed by H_SAVE_DISK for execution, the following option must be used at linking time of the load module: VACSEG=(SHARE=+10). In addition, one type-2 vacant entry in STN 1, 2 or 3 must be reserved, for example: SEGTAB3=(SHRLEVEL=2,VSEG=1).

For more information, refer to the following documents:

- *Data Management Utilities User's Guide*, VOLSAVE utility.
- *GPL System Primitives Reference Manual*, H_FILSAVE primitive.

4.1.15 H_SAVE_FILE

Function

Saves a physical copy of a disk file as a named component of a UFAS sequential disk file or tape file.

Syntax

```
$H_SAVE_FILE
          infile
          ,outfile
          [,SAVEMODE = i_char1]
          [,SAVENAME = i_char44]
          [,EXPORT = i_bit1]
          [,OUTALC = i_bit1]
          [,CATNOW = i_bit1]
          [,FMEDIA = i_bit1]
          [,DYNAMIC = i_bit1]
          [,NWRJOR = i_bit1] ;
```

Parameters

infile	specifies the internal file name (declared by H_FD) of the input disk file to be saved.
outfile	specifies the internal file name (declared by H_FD) of the output file that is to be used as SAVE file. The output file must be either a UFAS sequential disk file or a tape file.
SAVEMODE	specifies whether the disk file image will be placed as the first disk image in the output file or appended at the end of the output file. "C": H_SAVE_FILE creates a "first" disk file image identified by a name (SAVENAME) in the output file, i.e., the file is assumed not to contain any existing or non-obsolete disk file images. If these exist, they are overwritten. "C" is the default value. "A": H_SAVE_FILE adds a new disk file image identified by a name (SAVENAME) at the end of the output file. If the output file already contains other saved disk file images, H_SAVE_FILE will leave these images unchanged and add the new disk file image to the end of the output file.
SAVENAME	specifies the name to be given to the disk file image to be saved in the output file. This name may be specified to retrieve the disk file image within the input file at restore time. When omitted, the name given to the disk file image in the output file is the name (external file name) of the input disk file.
EXPORT	controls the interface that H_SAVE_FILE has with the After Journal. "1"B: do not reset to empty the set of after images for the file to be saved.

	"0"B: reset to empty the set of after images for the file to be saved (default value).
OUTALC	indicates whether or not the missing SAVE file is to be dynamically allocated. "1"B: dynamically allocates missing SAVE file before the save takes place. "0"B: uses the existing SAVE file (default value).
CATNOW	indicates whether or not the SAVE file is to be cataloged during dynamic allocation. "1"B: the output file is to be cataloged during dynamic allocation. OUTALC must be set to "1"B, and FILESTAT="04"X (UNCAT) must be specified for the output file in H_ASSIGN. "0"B (default): the SAVE file is not to be cataloged during dynamic allocation.
FMEDIA	for a cataloged SAVE file, specifies whether or not the existing cataloged list of volumes is to be erased and replaced by the volumes (given in H_ASSIGN) onto which dynamic allocation took place. "1"B: replace in the catalog, for the SAVE file, the existing list of volumes. OUTALC="1"B and CATNOW="1"B must also be specified. "0"B (default): do not replace in the catalog, for the SAVE file, the list of volumes. The file level catalog entry must be empty.
DYNAMIC	indicates whether or not the file to be saved may be shared with other concurrent steps. "1"B: infile may be shared with other concurrent steps. "0"B: exclusive access is required for infile (default value).

NWRJOR	indicates whether or not H_SAVE_FILE may write messages on the Job Occurrence Report (JOR). "1"B: no message is written to the JOR. "0"B: messages are written to the JOR (default value).

Normal return codes

DONE:	H_SAVE_FILE is successful. Infile is saved.
DONEID:	H_SAVE_FILE is successfully completed but the user specified that irrecoverable read errors were to be skipped over (ERROPT=SKIP in the infile H_FD). Such errors were encountered and several parts of infile have not been saved. The integrity and consistency of the saved file image is not assured.

Abnormal return codes

ARGERR:	one or more given parameters are wrong: infile is not a disk file, outfile is neither a tape file nor an existing UFAS sequential disk file, infile is a DUMMY file, etc...
ARVIOL:	access rights violation. The user does not have the required access rights on either the input or the output file.
CATERR:	assignment of either the input file or the output file contradicts their actual cataloged or uncataloged file status.
CONFLICT:	at least one of the outfile parameters has an incorrect value.
DATANAV:	the input file is in an unstable state.
DVNAV:	there are not enough devices (disk or tape drives) currently available.

EFNUNKN:	the input file or the output file does not exist on the specified volumes. OUTALC="1"B may be used for a missing output file.
EXHAUST:	the end of outfile has been reached and OUTALC="1"B has not been specified for extension or there is no more free space on the media list that support outfile.
IFNERR:	either the infile or the outfile parameter is wrong; it leads to invalid control structures.
IFNNASG:	at least one of the files, input or output is not assigned. Both files should be assigned.
IOFAIL:	an irrecoverable read error was encountered either on the input file or on the output file.
MDNAV:	a volume mounting request has been canceled by the operator.
MDERR:	incorrect or partial media list given for infile or outfile.
NOTOBS:	the outfile tape to be overwritten is still valid, the expiry date has not been reached.
NOWAIT:	synchronization error. Infile is being accessed by another task.
OPEN:	at least one of the files, infile or outfile is open. All these files should be closed before H_COPY_FILE is issued.
RECNFD:	a physical was missing when reading infile and the ERROPT parameter in H_FD requested a fatal error.
TRUNC:	refer to the EXHAUST return code.
RESNAV:	resource not available. DYNAMIC="1"B has been specified requesting a dynamic save but the dynamic save server has not been started.

SPACENAV:	there is not enough available space on the list of volumes specified in H_ASSIGN to allocate the output file.
YETALLOC:	the catalog entry referenced by the output file already contains a list of volumes. FMEDIA="1"B must be used to continue.

Some other return codes may also be returned to the caller especially when OUTALC="1"B is specified to request dynamic allocation of the missing output file (refer to H_PREALLOC in the *GPL System Primitives Reference Manual*).

Comments

- H_SAVE_FILE saves a disk file (as referenced by the infile parameter) onto a UFAS sequential disk or tape file (as referenced by the outfile parameter). The file resulting from a H_SAVE_FILE operation may be used as infile by the H_RESTORE_FILE primitive.
- Both infile and outfile must be assigned and closed when H_SAVE_FILE is called.
- To get a cataloged outfile, both CATNOW="1"B and OUTALC="1"B must be specified and FILESTAT="04"X (UNCAT) must be specified in H_ASSIGN. When the H_SAVE_FILE operation is over, the outfile's status changes from "04"X (UNCAT) to "00"X (CAT).
- If outfile is an existing UFAS sequential disk file, it must have been allocated with RECSIZE=6132 and RECFORM=V.
- Note that the file format used to save the disk file is the same as the one created by FILSAVE JCL utility or SAVE_FILE, SAVE_FILESET, SAVE_CATALOG GCL commands and H_FILSAVE GPL primitive so, it may be used as input file in a further FILREST, RESTORE_FILE, RESTORE_FILESET, RESTORE_CATALOG or H_FILREST operation.
- In order to reserve resources needed by H_SAVE_FILE for execution, the following option must be used at linking time of the load module: VACSEG=(SHARE=+10). In addition, one type-2 vacant entry in STN 1, 2 or 3 must be reserved, for example: SEGTAB3=(SHRLEVEL=2,VSEG=1).

- Generally, H_SAVE_FILE requires exclusive access to both infile and outfile. However, if DYNAMIC = "1"B is specified, then infile may be shared with concurrent steps. Dynamic save of shared files is fully described in the *Data Management Utilities User's Guide*.
- Whenever a malfunction occurs during the execution of an H_SAVE_FILE operation, the malfunction is reported to the user via JOR messages (except when NWRJOR="1"B is specified) and via return codes. The step completion code is then forced to SEV1 regardless of its previous value.

For more information, refer to the following documents:

- *GPL System Primitives Reference Manual*,
- *Data Management Utilities User's Guide*.

4.2 Volume-Level Primitives

4.2.1 H_DCEXTDESC

Function

Declares the structure for the file extents or volume free extents. This declarative provides a based structure used to map the output of the primitive H_READVTOC used with the LABEL="06"X, "07"X or "18"X option on FBO disks.

Syntax

```
$H_DCEXTDESC
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is H_EXT_DESC_.
ATTRIB	Storage class attribute. The only possible value is 'BASED (PTR)'.
NLVL1	Used to hide the level one sub-structure name of the declarative structure. Level 1 of this structure must be based with a default base. Data declaration *must not* follow H_DCEXTDESC to complete the Level 1 structure.

Comments

The H_DCEXTDESC declarative is used to map either the file extents description or the volume free extents description. It must only be used with ATTRIB='BASED(ptr)' storage class attribute.

In the case where the volume free extents are requested, then all these volume free extents are considered as defining a pseudo-file with a number of extents equal to the number of free extents and with a file block size equal to the volume data block size.

Therefore, the following H_DCEXTDESC subfields have the meanings as indicated:

- prefix_TOTAL_FB_FB31 is the total number of free volume data blocks,
- prefix_DB_SZ_FB31 is the volume data block size,
- prefix_DB_PER_FB_FB15 equals 1,
- prefix_NBEXT_LB16 is the number of free volume extents,
- prefix_EXTENT_TYPE_B8 equals "01"X,
- prefix_FIRST_DB_LB32 is the number, within the volume, of the first data block of the corresponding free extent. Note that the number of the first volume data block is 0.
- prefix_LAST_DB_LB32 is the number, within the volume, of the last data block of the corresponding free extent.

Note that a disk volume may have a great number of free extents depending on space fragmentation but a file may only have 16 extents at most per disk volume.

For more information, refer to the H_READVTOC primitive, OUTAREA parameter.

Expansion

The declaration generated by the macro expansion of $H_DCEXTDESC PREFIX='B_', ATTRIB='BASED(SEG_PTR)' is as follows:

```
/*    $H_DCEXTDESC PREFIX=B_, ATTRIB=BASED(SEG_PTR);  */

DCL  1  H_EXT_DESC              BASED(SEG_PTR),
     2  B_FIXED_PART,
     3  B_VERSION_LB8           LOGBIN(8),     /* Version number. First version is 0     */
     3  B_VSN_C6                CHAR (6),      /* Volume Serial Number.                  */
     3  B_VSEQN_LB16            LOGBIN(16),    /* Volume Sequence Number.                */
                                               /* Identifies the order of physical       */
                                               /* volumes for first allocation           */
                                               /* on this volume.                        */
     3  B_DVCLASS,
     4  B_DVTYPE_C2             CHAR(2),
     4  B_DVATTR_B16            BIT(16),
     3  B_TOTAL_FB_FB31         FIXED BIN(31), /* Number of  file blocks on this volume.  */
     3  B_DB_SZ_FB31            FIXED BIN(31), /* Length of the DATA BLOCK  (A file must  */
                                               /* be allocated on only 1 partition).      */
     3  B_TOTAL_NB_CYL_LB16     LOGBIN(16),    /* Nb of cylinders. Used for FBO_CKD.      */
     3  B_NB_TRK_PER_CYL_FB15   FIXED BIN(15), /* Nb of tracks per cylinder. Used         */
                                               /* for FBO_CKD                             */
     3  B_NB_DB_PER_FB_FB15     FIXED BIN(15), /* Nb of DATA BLOCKS per FILE BLOCK        */
     3  B_DB_PER_TRACK_FB15     FIXED BIN(15), /* Nb of DATA BLOCKS per track. Useful     */
                                               /* information for FBO_CKD only.           */
     3  B_RPS_DB_ON_FBO_CKD,                   /* Array of 128 bytes  (Maximum value      */
                                               /* for RPS). For FBO_CKD only, this field  */
                                               /* will be filled.                         */
     4  B_RPS_NB_LB8 (128)      LOGBIN(8),
     3  B_FL_BLK_GROUP_SZ_FB15  FIXED BIN(15), /* For FBO_CKD: Optimum data length        */
                                               /* transfer; only available if there       */
                                               /* is an integer nb of FB per track,       */
                                               /* otherwise 0.                            */
                                               /* For FBO_FSA: Must be 0.                 */
     3  B_NBEXT_LB16            LOGBIN(16),    /* Number of extents (limited to 16)       */
                                               /* on this volume.                         */
     3  *                       BIT(16),
     3  B_RFU_C6                CHAR(6),       /* Must be (6) "00"H.                      */
     3  B_REAL_NBEXT_LB16       LOGBIN(16),    /* Real number of extents on this volume   */
     2  B_VARIABLE_PART,
     3  B_EXT_ARRAY (B_NBEXT_LB16),
     4  B_EXTENT_TYPE_B8        BIT(8),        /* Indicates the category of the           */
                                               /* records for which this extent           */
                                               /* is reserved or the processing mode      */
                                               /*   (Form: bit(8))                        */
                                               /*   "00"X: subsequent  bytes              */
                                               /*           are meaningless               */
                                               /*   "01"X: data records                   */
                                               /*   "02"X: overflow records               */
                                               /*   "04"X: index records                  */
                                               /*   "E3"X: temporary extent               */
                                               /*   other values forbidden                */
     4  B_FIRST_AM_FB_LB24      LOGBIN(24),    /* FILE BLOCK nb of the first FILE         */
                                               /* BLOCK in the extent.                    */
```

```
4  B_LAST_AM_FB_LB24       LOGBIN(24),    /* FILE BLOCK nb of the last  FILE         */
                                          /* BLOCK in the extent.                    */
4  B_EXT_FLAGS,
5  B_LAST_EXT_B1           BIT(1),        /* "1"B -> Last extent allocated,          */
                                          /* all volumes merged.                     */
5  B_END_VOL_DATA_B1       BIT(1),        /* "1"B -> This extent contains the        */
                                          /* logical end of this file.               */
5  B_LAST_PERMEXT_B1       BIT(1),        /* "1"B = Temporary extents exist after    */
                                          /* this extent, but this one is the last   */
                                          /* permanent extent, all volumes merged.   */
5  B_RFU_B5                BIT(5),        /* Must be "00000"B .                      */
4  B_RFU_C4                CHAR(4),       /* Must be (4) "00"H.                      */
4  B_EXTENT_DESC,
5  B_PARTITION_NUMBER_LB8  LOGBIN(8),     /* Partition Nb including this extent      */
                                          /* Always zero in this version             */
                                          /* Wanted redundancy for future use.       */
5  B_FIRST_DB_LB32         LOGBIN(32),    /* Number of the data block  which         */
                                          /* is included into the first file         */
                                          /* block of this extent.                   */
5  B_LAST_DB_LB32          LOGBIN(32),    /* Address of last allocated data block.   */
                                          /* Do not use to compute the file block    */
                                          /* address.  Nb of last data blocks in     */
                                          /* this extent (one file block is mapped   */
                                          /* by n DATA BLOCKS).                      */
5  B_RFU_C1                CHAR(1);       /* Must be (1) "00"H.                      */
```

4.2.2 H_DCEXTVOL

Function

Declares the structure for the volume label. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="08"X option.

Syntax

```
$H_DCEXTVOL
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is H_EXT_VOL_.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DCEXTVOL is as follows:

```
/*   $H_DCEXTVOL;  */

DCL 1  H_EXT_VOL,
    2  H_EXT_VOL_VOLABID_C3         CHAR(3),      /* Identifier. Contains "VOL"           */
    2  H_EXT_VOL_VOLABNB_C1         CHAR(1),      /* Defines the type of the VTOC         */
                                                  /* "1" = Old VTOC --> VBO disk          */
                                                  /* "2" = New VTOC --> FBO disk          */
    2  H_EXT_VOL_VSN_C6             CHAR(6),      /* Volume Serial Number                 */
    2  H_EXT_VOL_OWNER_C14          CHAR(14),     /* Contains the name of the OWNER of    */
                                                  /* the disk                             */
    2  H_EXT_VOL_DVCLASS,                         /* Identifies the device class of the   */
                                                  /* disk                                 */
    3  H_EXT_VOL_DVTYP_C2           CHAR(2),
    3  H_EXT_VOL_DVATTR_B16         BIT(16),      /* Device attribute of the disk.        */
    2  H_EXT_VOL_VALGCOS_C6         CHAR(6),      /* This field is built from system base */
                                                  /* It contains the OS_NAME.             */
    2  H_EXT_VOL_SYSCODE_C3         CHAR(3),      /* Contains version of PREPARE_VOLUME   */
                                                  /* utility                              */
                                                  /* "000"  Before R5                     */
                                                  /* "002"  R5M                           */
                                                  /* "003"  Between V1 release included   */
                                                  /*        and first FBO release         */
                                                  /*        excluded.                     */
                                                  /* "004"  First FBO release.            */
    2  H_EXT_VOL_PREPDATE,                        /* Indicates the preparation date of    */
                                                  /* the volume in form YDDD              */
    3  H_EXT_VOL_YP_LB8             LOGBIN(8),    /* Relative displacement / 1900.        */
    3  H_EXT_VOL_DDDP_LB16          LOGBIN(16),   /* From 1 to 366                        */
    2  H_EXT_VOL_DISKIND,
    3  H_EXT_VOL_PROTECTED_VOL_B1   BIT(1),       /* "1"B -> Volume is protected          */
    3  H_EXT_VOL_UNSTABLE_B1        BIT(1),       /* "1"B -> Volume is unstable because   */
                                                  /* it has not been fully restored.      */
    3  H_EXT_VOL_FORMATED_B1        BIT(1),       /* "1"B-> Volume is formatted but the   */
                                                  /* VTOC files are still not built       */
    3  H_EXT_VOL_QUOTA_ACTIVE_B1    BIT(1),       /* "1"B-> Volume is submitted to        */
                                                  /* quota management.                    */
    3  H_EXT_VOL_DISKIND_RFU2_B12   BIT(12),
    2  H_EXT_VOL_VBO_DISK_PART,
    3  H_EXT_VOL_NBCYL_LB16         LOGBIN(16),   /* Number of available cylinders on     */
                                                  /* disk except  cylinders containing    */
                                                  /* alternate tracks.                    */
                                                  /* MS/D500      : 707                   */
    3  H_EXT_VOL_NBTRK_LB16         LOGBIN(16),   /* Number of tracks per cylinder        */
                                                  /* MS/D500      : 24                    */
    3  H_EXT_VOL_DEPLV_PART,
    4  H_EXT_VOL_NBBYT_LB24         LOGBIN(24),   /* Max Data Length possible for a       */
                                                  /* non-keyed record                     */
                                                  /* MS/D500      : 29013                 */
    4  H_EXT_VOL_DEVIND_LB8         LOGBIN(8),    /* Status Byte                          */
                                                  /* MS/D500      : 16                    */
    4  H_EXT_VOL_KEYNLAST_LB16      LOGBIN(16),   /* Overhead bytes for a keyed           */
```

```
                                        /* physical block which is not the       */
                                        /* last of the track.                    */
                                        /* Tolerance factor must be added if     */
                                        /* STATUS = 2                            */
                                        /* Must be multiplied by 4 if            */
                                        /* if STATUS = 16                        */
                                        /* MS/D500      : 126                    */
  4  H_EXT_VOL_KEYLAST_LB16      LOGBIN(16),   /* Overhead bytes for the last keyed     */
                                        /* block of the track.                   */
                                        /* Tolerance factor must be added        */
                                        /* if STATUS = 2                         */
                                        /*  Must be multiplied by 4              */
                                        /* if STATUS = 16                        */
                                        /* MS/D500      : 126                    */
  4  H_EXT_VOL_NOKEY_LB16        LOGBIN(16),   /* Overhead bytes to be subtracted       */
                                        /* from KNLAST & KLAST when there is     */
                                        /* no key field                          */
                                        /* . Tolerance factor must be  added     */
                                        /* if STATUS = 2                         */
                                        /* Must be multiplied by 4               */
                                        /* if STATUS = 16                        */
                                        /* MS/D500      :  43                    */
  4  H_EXT_VOL_DEVTOL_LB16       LOGBIN(16),   /* Tolerance Factor or auditing          */
                                        /* factor                                */
                                        /* MS/D500      :  0                     */
  4  H_EXT_VOL_ALSTARTCC_LB16    LOGBIN(16),   /* Cylinder number which contains        */
                                        /* the first alternate track.            */
  4  H_EXT_VOL_NBALTRK_LB16      LOGBIN(16),   /* Number of free alternate tracks       */
  4  H_EXT_VOL_ALTRK,                         /* Next alternate track address.         */
                                        /* Form CC HH                            */
  5  H_EXT_VOL_CCA_LB16          LOGBIN(16),
  5  H_EXT_VOL_HHA_LB16          LOGBIN(16),
  4  H_EXT_VOL_ALNBCYL_LB8       LOGBIN(8),    /* Number of cylinders reserved for      */
                                        /* alternate tracks on this volume       */
  4  H_EXT_VOL_LGHAR0_LB16       LOGBIN(16),   /* Length in bytes of the HA, gaps & R0  */
                                        /* MS/D500      :  480                   */
  2  H_EXT_VOL_DSMGT_PART,
  3  H_EXT_VOL_PARTITION_LB8     LOGBIN(8),    /* Partition containing the VTOC,        */
                                        /* always the first in the disk          */
  3  H_EXT_VOL_VTOCFBSIZE_LB16   LOGBIN(16),   /* Size of VTOC (S and P) s CIs          */
  3  H_EXT_VOL_VTOCADDR_LB32     LOGBIN(32),   /* Data block number of the first        */
                                        /* DFL01 in VTOCS relative to the        */
                                        /* beginning of the partition 0.         */
  3  H_EXT_VOL_VTOC_NB_EFN_FB31  FIXED BIN(31),/* Nb of EFN asked at VTOC creation      */
                                        /* time                                  */
  3  H_EXT_VOL_SITE_B32          BIT(32),      /* Physical site on which the            */
                                        /* site.quota has been created.          */
  3  H_EXT_VOL_DB_SZ_PART_0_FB15 FIXED BIN(15),/* Data Block size of the partition 0    */
  3  H_EXT_VOL_NB_DB_PART_0_LB32 LOGBIN(32),   /* Number of DB into the data            */
                                        /* partition 0. For CKD disks, this      */
                                        /* number does not included the DB       */
                                        /* of the alternate area                 */
  3  H_EXT_VOL_NB_DB_PER_TRK_LB8 LOGBIN(8),    /* Number of DB per track in the         */
                                        /* partition 0.  (Used at ISL time).     */
  3  H_EXT_VOL_RFU_C21           CHAR(21);     /* Must be (21) "00"H.                   */
```

4.2.3 H_DCLOGLAB

Function

Declares the structure for the file logical label. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="01"X option.

Syntax

```
$H_DCLOGLAB
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is "H_LOG_LAB_".
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DCLOGLAB PREFIX='LOG_LAB_' is as follows:

```
/*            $H_DCLOGLAB PREFIX=LOG_LAB_;            */
DCL   1  H_LOG_LABEL  AUTOMATIC,
      2  LOG_LAB_LABEL_TYPE_B8     BIT(8),      /* "01"X -> Main file label record        */
      2  LOG_LAB_LABEL_VERSION_LB8 LOGBIN(8),   /* "00"X -> First Version.Compatible      */
                                                /*          with the old label 1.         */
                                                /* "01"X -> R7                            */
                                                /* "02"X -> FBO                           */
      2  LOG_LAB_LVNAME_C16        CHAR(16),    /* Logical Volume Name (RFU)              */
                                                /* Must be (16) "00"H.                    */
      2  LOG_LAB_EFN_C44           CHAR(44),    /* External File Name                     */
      2  LOG_LAB_VSN_C6            CHAR(6),     /* Volume serial number of the first      */
                                                /* volume on which the file resides       */
      2  LOG_LAB_VSEQN_LB16        LOGBIN(16),  /* Volume SEQuence Number identifies the  */
                                                /* order of the physical volumes for the  */
                                                /* first allocation on this volume        */
      2  LOG_LAB_CRE_DATE,                       /* Creation date of the file in form:     */
                                                /* YDDD  (Year, Day in the year)          */
      3  LOG_LAB_CRE_Y_LB8         LOGBIN(8),   /* Relative displacement / 1900.          */
      3  LOG_LAB_CRE_DD_LB1        LOGBIN(16),  /* From 1 to 366                          */
      2  LOG_LAB_EXP_DATE,                       /* Expiration date of the file in         */
                                                /* format YDDD                            */
                                                /*   (Form: LOGBIN(8),LOGBIN(16))         */
      3  LOG_LAB_EXP_Y_LB8         LOGBIN(8),   /* Relative displacement / 1900.          */
      3  LOG_LAB_EXP_DD_LB16       LOGBIN(16),  /* 1  to 366                              */
      2  LOG_LAB_SYSCODE_C13       CHAR(13),    /* Identifies the operating system that   */
                                                /* created the file. This field is built  */
                                                /* from system base. It contains:         */
                                                /* Bytes 1 to 7: OS_NAME.                 */
                                                /* Bytes 8 to 9: EXT_RLS_NAME.            */
                                                /* Bytes 11 to 13: version number.        */
                                                /* (First version number will be 004)     */
                                                /* (This field contains the version       */
                                                /* of PREPARE_VOLUME utility).            */
                                                /* "000" = Before R5.                     */
                                                /* "002" = R5M.                           */
                                                /* "003" = Between V1 release include     */
                                                /* and first FBO release excluded         */
                                                /* "004" = First FBO release.             */
                                                /* (Form:13 alphanumeric characters)      */
      2  LOG_LAB_FILORG_B8         BIT(8),      /* Specifies the file organization        */
                                                /* and relocatability                     */
                                                /* (Form: BIT(8))                         */
                                                /* 0000 000x - No file organization       */
                                                /* 1000 000x - indexed sequential         */
                                                /* 0100 000x - sequential                 */
                                                /* 0001 000x - IDS                        */
                                                /* 0000 100x - Linked queued              */
                                                /* 0000 001x - UFAS relative              */
                                                /* 0000 0001 - UFAS random                */
                                                /* other values - forbidden               */
```

```
2  LOG_LAB_FILEFORM_B8       BIT(8),      /* Specifies the file format family       */
                                          /* (Form: BIT(8))                         */
                                          /* 1000 0000 - BFAS                       */
                                          /* 0000 0100 - UFAS                       */
                                          /* 0000 0000 - no file                    */
                                          /* organization (NONE)                    */
                                          /* other values - forbidden               */
2  LOG_LAB_FILIND,
3  LOG_LAB_END_VOL_DATA_B1   BIT(1),      /* "1"B -> The logical end of this        */
                                          /* file is on this volume.                */
                                          /* See END_VOL_DATA_B1 of H_DCEXTDESC     */
                                          /* to find the last extent where the      */
                                          /* data is recorded.                      */
3  LOG_LAB_LAST_VOLUME_B1    BIT(1),      /* "1"B -> The last allocated extent      */
                                          /* is on this volume.                     */
3  LOG_LAB_UNSTABLE_B1       BIT(1),      /* "1"B -> File unstable because it       */
                                          /* is not fully restored.                 */
3  LOG_LAB_CATALOGUED_B1     BIT(1),      /* "1"B -> The file is cataloged          */
3  LOG_LAB_RFU1_B4           BIT(4),      /* Must be "0000"B                        */
2  LOG_LAB_FILE_VERSION_C2   CHAR(2),     /* Contains the file version number       */
                                          /* Form: 2 numerical characters           */
2  LOG_LAB_OPTCODE,
3  LOG_LAB_WR_CHECK_B1       BIT(1),      /* "1"B -> Writing validity check         */
                                          /* required                               */
3  LOG_LAB_FIXNBLK_B1        BIT(1),      /* "1"B -> Fixed number of blocks         */
                                          /* per track                              */
3  LOG_LAB_MASTER_B1         BIT(1),      /* "1"B -> Master index exists.           */
                                          /* BFAS IND only.                         */
                                          /* Meaningless in FBO.                    */
3  LOG_LAB_GENOV_B1          BIT(1),      /* "1"B -> General Overflow exists.       */
                                          /* BFAS IND only.                         */
                                          /* Meaningless in FBO.                    */
3  LOG_LAB_CYLOV_B1          BIT(1),      /* "1"B -> Cylinder Overflow exists.      */
                                          /* BFAS IND only.                         */
                                          /* Meaningless in FBO.                    */
                                          /* This RFU maps the location of VBO      */
                                          /* flags.                                 */
                                          /* It must be equal to "0000"B.           */
3  LOG_LAB_USER_MODIFIED_B1  BIT(1),      /* "1"B -> The maintenance tool has       */
                                          /* modified this file label.              */
3  LOG_LAB_DEL_REC_B1        BIT(1),      /*"1"B -> Don't check record deletion     */
3  LOG_LAB_RFU_B1            BIT(1),      /* Must be equal to "0"B                  */
2  LOG_LAB_PHYRCDSZ_LB24     LOGBIN(24),  /* Contains the file block size           */
                                          /* (for UFAS) rounded up to a             */
                                          /* multiple of 512 bytes for UFAS         */
                                          /* or the BLKSIZE for QUEUED.             */
                                          /* (Unit=byte, form: LOGBIN(24))          */
2  LOG_LAB_RCDFORM_B8        BIT(8),      /* Specifies the record format in the file*/
                                          /* (Form: BIT(8))                         */
                                          /* 100x x000 - fixed length               */
                                          /* 010x x000 - variable length            */
                                          /* 1100 x000 - undefined length           */
                                          /* xx01 x000 - blocked records            */
2  LOG_LAB_RCDSZ_LB24        LOGBIN(24),  /* Maximum length of the                  */
                                          /* logical record in the file             */
                                          /* (Form: LOGBIN(24))                     */
```

```
2  LOG_LAB_KEYSZ_LB8          LOGBIN(8),   /* Contains the length of the key         */
                                           /* portion of the record                  */
                                           /* (Unit=byte, form: LOGBIN(8))           */
2  LOG_LAB_KEYLOC_LB24        LOGBIN(24),  /* Relative location of the key           */
                                           /* within the record                      */
                                           /* (Unit=byte, form: LOGBIN(24))          */
2  LOG_LAB_BLOCKSZ_LB16       LOGBIN(16),  /* Contains the FB size asked by          */
                                           /* the user for UFAS,                     */
                                           /* or the directory size for QUEUED.      */
2  LOG_LAB_PHRCDBLK_B8        BIT(8),      /* For QUEUED contains the directory type */
                                           /* For UFAS contains CI free space.       */
2  LOG_LAB_FILCHAR,
3  LOG_LAB_FL_SECT_EMPTY_B1   BIT(1),      /* "1"B ->File section is empty           */
                                           /* on this volume                         */
                                           /* - Meaningful when VERSION >= "001"     */
                                           /* - Must Be Zero when VERSION < "001"    */
3  LOG_LAB_RFU_B3             BIT(3),      /* Must be "000"B                         */
3  LOG_LAB_COMPACT_B1         BIT(1),      /* "1"B -> File is compacted              */
3  LOG_LAB_RFU1_B3            BIT(3),      /* Must be "000"B.                        */
2  LOG_LAB_PARAM_B8           BIT(8),      /* Indicates in which way                 */
                                           /* space is allocated to this file        */
                                           /* (Form: bit(8))                         */
                                           /* 0000 000x - in tracks relative to      */
                                           /*             a specific location        */
                                           /* 0100 000x - in FILE BLOCKs             */
                                           /* 1100 000x - in cylinders.              */
                                           /* 1000 000x - in tracks.                 */
2  LOG_LAB_INCRSZ_LB24        LOGBIN(24),  /* Specifies the increment size           */
                                           /* in the unit which has been             */
                                           /* used for file creation                 */
                                           /* (Form: LOGBIN(24))                     */
2  LOG_LAB_STARTBLK,
3  LOG_LAB_TTS_LB24           LOGBIN(24),
3  LOG_LAB_RS_LB8             LOGBIN(8),   /* - for linked queued, TTS_LB24          */
                                           /* represents the logical track size      */
                                           /* logical track size in LOGBIN(16)       */
                                           /* BAM type in LOGBIN(8)                  */
                                           /* RS_LB8 must be 0                       */
2  LOG_LAB_LASTBLK,                        /* Indicates the location of the          */
                                           /* last physical block of a BFAS          */
                                           /* sequential file.                       */
                                           /* The format is TTTRLL where TTT         */
                                           /* is the relative track address          */
                                           /* containing the last physical block     */
                                           /* (Form: LOGBIN(24))                     */
                                           /* R is the logical physical block        */
                                           /* number in the track                    */
                                           /* (Form: LOGBIN(8))                      */
                                           /* and LL is the number of bytes          */
                                           /* remaining on the track                 */
                                           /* following the last physical block.     */
                                           /* (Form: LOGBIN(16))                     */
3  LOG_LAB_TTT_LB24           LOGBIN(24),  /* For a UFAS file, the first byte         */
                                           /* contains the number of user's labels   */
3  LOG_LAB_R_LB8              LOGBIN(8),
3  LOG_LAB_LL_LB16            LOGBIN(16),
```

```
2  LOG_LAB_RCDBLOC_B8        BIT(8),     /* Key/Data blocks word round-up          */
                                         /* for SMD only                           */
2  LOG_LAB_FILE_SHARING(8),              /* Contains access and sharing            */
                                         /* conditions encountered on 8 possibly   */
                                         /* shared systems.                        */
3  LOG_LAB_MULTISHR_B5       BIT(5),     /* Multi systems concurrent access        */
                                         /* permission                             */
                                         /* "00000"B -> Normal                     */
                                         /* "10000"B -> None                       */
                                         /* "10001"B -> Onewrite                   */
                                         /* "10011"B -> Free                       */
3  LOG_LAB_ACCESS_B3         BIT(3),     /* "000"B -> Clear (This file is no       */
                                         /* longer used by this computer.          */
                                         /* "010"B -> Read only                    */
                                         /* "100"B -> Write (and/or read)          */
                                         /* "110"B -> Read without writer          */
                                         /* "001"B -> Exclusive access             */
2  LOG_LAB_RFU_DIRTY_READ_B8 BIT(8),     /* Must be "00000000"B.                   */
2  LOG_LAB_RFU_C6            CHAR(6);    /* Must be (6)"00"H.                      */
```

4.2.4 H_DCMSINFO

Function

Declares the structure MSINFO referenced as output by the H_MSINFO primitive.

Syntax

```
$H_DCMSINFO
          [,PREFIX = l_identifier22]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DCMSINFO; is as follows:

```
DCL 1 MSINFO,

     2  VTOC_TYPE     BIT (1),
           /* "0"B --> OLD VTOC FORMAT VBO ONLY            */
           /* "1"B --> NEW VTOC FORMAT FBO DISK            */
     2  DISK_TYPE     BIT (1),
           /* "0"B --> FSA DISK                            */
           /* "1"B --> CKD DISK                            */
     2  RESIDENT      BIT (1),
           /* "1"B --> RESIDENT VOLUME                     */
     2  BKST          BIT (1),
           /* "1"B --> BACKING STORE DISK                  */
     2  SYSTEM        BIT (1),
           /* "1"B --> SYSTEM DISK                         */
     2  TEMP          BIT (1),
           /* "1"B --> TEMPRY FILES AUTHORIZED IF RESIDENT */
     2  SHARABLE      BIT (1),
           /* "1"B --> SHARED DISK                         */
     2  QUOTA         BIT (1),
           /* "1"B --> QUOTA ACTIVE                        */
     2  MIRROR        BIT (1),
           /* "1"B --> MIRRORED DISK SEE H_DCMSINFO_MIRROR */
     2  FORMATED      BIT (1),
           /* "1"B --> DISK JUST FORMATED (NO VTOC)        */
```

```
2  RAID1          BIT(1),
      /* "1"B --> RAID1 DISK                          */
2  RAIDS          BIT(1),
      /* "1"B --> RAIDS DISK                          */
2  VALUE_MBZ      BIT (20);
```

4.2.5 H_DCMSINFO_MIRROR

Function

Declares the structure MSINFO_MIRROR referenced as output by the H_MSINFO primitive.

Syntax

```
$H_DCMSINFO_MIRROR

          [,PREFIX = l_identifier18]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DCMSINFO_MIRROR; is as follows:

```
DCL 1 MSINFO_MIRROR,

     2  PRIMARY      CHAR (4),
           /* DVNAME OF PRIMARY DISK                          */
     2  SECONDARY    CHAR (4),
           /* DVNAME OF SECONDARY DISK                        */
     2  VALID        BIT (1),
           /* "1"B --> VALID MIRRORED DISK                    */
     2  ALONE        BIT (1),
           /* "1"B --> MIRRORED DISK ALONE                    */
     2  REFILLING    BIT (1),
           /* "1"B --> PRIMARY BEING COPIED ONTO SECONDARY */
     2  DEFERRED     BIT (1),
           /* "1"B --> REFILLING DEFERRED                     */
     2  VALUE_MBZ    BIT (28);
```

NOTE:

The H_DCMSINFO_MIRROR structure information is valid only if the MIRROR bit of the H_DCMSINFO structure is set.

4.2.6 H_DCVOLINFO

Function

Declares the structure used as an output parameter by the H_GETVOLINFO primitive.

Syntax

```
$H_DCVOLINFO
          [NAME = l_identifier15]
          [,PREFIX= l_identifier16]
          [,ATTRIB= l_char] ;
```

Parameters

NAME	gives the name of the level 1 of the structure. Default value is H_VOLINFO.
PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value is 'AUTOMATIC'.

Expansion

The declaration generated by the macro expansion of $H_DVOLINFO is as follows:

```
DCL  1  H_VOLINFO,
     2  LABEL              CHAR (6),
     2  VOLSER             CHAR (6),
     2  STATUS,
     3  STANDARD           BIT (1),
     3  NON_STANDARD       BIT (1),
     3  WORK               BIT (1),
     3  WRITE_PROTECT      BIT (1),
     3  *                  BIT (4),
     3  INCOHERENT         BIT (1),
     3  *                  BIT (7);
```

Field Description

LABEL	External label.
VOLSER	Volume Serial Number in the VOL1 label.
STANDARD	"1"B –> the volume is standard.
NON_STANDARD	"1"B –> the volume is non-standard.
WORK	"1"B –> the volume is WORK.
WRITE_PROTECT	"1"B –> the volume is write protected.
INCOHERENT	"1"B –> the volume is incoherent. The following are examples of "incoherent" volumes: – cartridge without optical label, – cartridge without magnetic label, – cartridge with different optical and magnetic labels, – WORK cartridge which is not SCRATCH for the server,

- WORK cartridge which does not follow name convention,
- SCRATCH cartridge which is not WORK for GCOS 7,
- cartridge with errors.

4.2.7 H_DFL1

Function

Declares the structure for the file label 1. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="06"X or LABEL="07"X option on VBO disks.

Syntax

```
$H_DFL1
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DFL1 is as follows:

```
DCL  1  FL1,
     2  FLNAME     CHAR(44),       /* FILE NAME                                          */
     2  KEYF1      BIT(8),         /* KEY EQUAL TO "F1"X INDICATING A FILE LABEL 1       */
     2  FSN        CHAR(6),        /* FILE SEQUENCE NUMBER:                              */
                                   /* VOLUME SERIAL NUMBER OF THE FIRST                  */
                                   /* VOLUME SUPPORTING THE FILE                         */
     2  VSEQN      BIT(16) ,       /* VOLUME SEQUENCE NUMBER:                            */
                                   /* SEQUENCE NUMBER OF THE PRESENT                     */
                                   /* VOLUME WITHIN THE FILE                             */
     2  CREDATE,                   /* CREATION DATE IN THE FORM YY/DDD                   */
     3  YC         BIT(8),
     3  DDC        BIT(16),
     2  EXPDATE,                   /* EXPIRATION DATE IN THE FORM YY/DDD                 */
     3  YE         BIT(8),
     3  DDE        BIT(16),
     2  NBEXT      BIT(8),         /* NUMBER OF EXTENTS THE FILE LIES ON, IN THAT VOLUME */
                                   /* THE USER LABEL EXTENT IS NOT                       */
                                   /* INCLUDED IN THAT NUMBER                            */
     2  RS1        BIT(8),         /* NOT USED                                           */
     2  RCDBLOC    BIT(8),         /* RECORD/BLOCK MAPPING FOR INTEGRATED ACCESS ONLY:   */
                                   /* XXXXX001    SERIAL                                 */
                                   /* XXXXX010    DIRECT                                 */
                                   /* XXXXX100    INDIRECT                               */
                                   /* XX1XXXXX    BLOCK HEADER INCLUDES                  */
                                   /*             THE BLOCK SERIAL NUMBER                */
                                   /* X1XXXXXX    UNDEFINED BLOCK LENGTH                 */
                                   /* 0XXXXXXX    FIXED BLOCK LENGTH                     */
                                   /* 1XXXXXXX    VARIABLE BLOCK LENGTH                  */
     2  SYSCODE    CHAR(13),       /* UNIQUE IDENTIFICATION OF THE OPERATING SYSTEM      */
     2  BLOCKSZ    BIT(16),        /* FOR INTEGRATED ACCESS:                             */
                                   /*   BLOCK SIZE,                                      */
                                   /* FOR QUEUED FILES                                   */
                                   /*   NUMBER OF UNUSED BYTES IN LAST DIRECTORY BLOCK   */
     2  PHRCDBLK   BIT(8),         /* NUMBER OF PHYSICAL RECORDS PER PAGE ( ONLY USED    */
                                   /* FOR INTEGRATED ACCESS)                             */
     2  FILCHAR    BIT(8),         /* FILE CHARACTERISTICS:                              */
                                   /* 1XXXXXX0    FILE SECTION EMPTY                     */
                                   /* XX1XXXX0    FORMATTED FILE                         */
                                   /* XXXX1XX0    COMPACTED FILE                         */
                                   /* XXXX0100    SERIAL FILE                            */
                                   /* XXXX0010    LINKED FILE                            */
     2  STARTBLK,                  /* FOR SEQUENTIAL FILE:                               */
                                   /*     ADDRESS OF THE FIRST BLOCK RELATIVE TO THE     */
                                   /*     BEGINNING OF THE FILE                          */
                                   /* FOR LINKED SEQUENTIAL FILE:                        */
                                   /*     RELATIVE BLOCK ADDRESS OF THE FIRST BLOCK      */
                                   /*     WITHIN THE GLOBAL EXTENT                       */
                                   /* FOR PARTITIONED QUEUED FILE:                       */
                                   /*     DIRECTORY BLOCK SIZE (IN TTS)                  */
     3  TTS        BIT(16),
     3  RS         BIT(8),
     2  FILORG     BIT(8),         /* FILE ORGANIZATION:                                 */
                                   /* 0000000X    NO FILE ORGANIZATION                   */
                                   /* 1000000X    COEXISTENT INDEXED SEQUENTIAL          */
                                   /* 0100000X    PHYSICAL SEQUENTIAL                    */
                                   /* 0010000X    DIRECT                                 */
                                   /* 0001000X    RANDOM                                 */
                                   /* 0000100X    LINKED QUEUED                          */
                                   /* 0000010X    PACKED INDEXED SEQUENTIAL              */
```

```
                                /* 0000001X    PARTITIONED QUEUED                    */
                                /* 0100100X    LINKED SEQUENTIAL                     */
                                /* 0011000X    INTEGRATED ACCESS                     */
                                /* 0101000X    CATALOG OR SUBCATALOG                 */
                                /* XXXXXXX1    UNRELOCATABLE                         */
  2  FILEFORM    BIT(8),        /* FILE FORMAT:                                      */
                                /* 10000000    LEVEL 64 (BFAS)                       */
                                /* 01000000    OS360                                 */
                                /* 00100000    DOS360                                */
                                /* 00010000    HFAS                                  */
                                /* 00001000    LEVEL 62                              */
                                /* 00000100    UFAS                                  */
                                /* 00000010    ANSI                                  */
                                /* 00000001    NONSTD                                */
  2  RCDFORM     BIT(8),        /* RECORD FORMAT:                                    */
                                /* 10XX0000   FIXED LENGTH RECORD                    */
                                /* 01XX0000   VARIABLE LENGTH RECORD                 */
                                /* 11XXX000   UNDEFINED LENGTH RECORD                */
                                /* XX1XX000   TRACK OVERFLOW                         */
                                /* XXX1X000   BLOCKED RECORDS                        */
                                /* 10XX1000   STANDARD BLOCK WITH NO TRUNCATED BLOCKS */
                                /*            OR UNFILLED TRACK EMBEDDED IN A FILE   */
                                /* 01XX1000   SPANNED RECORDS                        */
  2  OPTCODE     BIT(8),        /* OPTION CODE:                                      */
                                /* 1XXXXXXX  WRITE VALIDITY CHECK IS REQUESTED       */
                                /* XXXXX1XX  DELETION OPTION IS REQUESTED            */
                                /* FOR SEQUENTIAL AND QUEUED FILES:                  */
                                /*  XXX1XXXX  FIXED LENGTH BLOCKS (LINKED QUEUED)    */
                                /*  XX1XXXXX  ADDRESS FORMAT IS RRR,                 */
                                /*            OTHERWISE IT IS TTR                    */
                                /*  X1XXXXXX  FIXED NUMBER OF PHYSICAL RECORDS       */
                                /*            PER TRACK                              */
                                /*  XXXXXX01  SYSTEM STANDARD FORMAT OF RECORDS      */
                                /*  XXXXXX10  PRINTER FORMAT (SARF)                  */
                                /* FOR INDEXED SEQUENTIAL:                           */
                                /*  XX1XXXXX  MASTER INDEXES EXIST                   */
                                /*  XXX1XXXX  INDEPENDENT OVERFLOW EXISTS            */
                                /*  XXXX1XXX  CYLINDER OVERFLOW AREA EXISTS          */
                                /* FOR DIRECT FILE:                                  */
                                /*  X1XXXXXX  TRACK OVERFLOW                         */
                                /*  XXXX1XXX  ACTUAL ADDRESSING                      */
                                /*  XXXXXXX1  RELATIVE ADDRESSING                    */
  2  PHYRCDSZ    BIT(16),       /* PHYSICAL RECORD SIZE                              */
  2  RCDSZ       BIT(16),       /* RECORD SIZE                                       */
  2  KEYSZ       BIT(8),        /* KEY SIZE                                          */
  2  KEYLOC      BIT(16),       /* KEY LOCATION WITHIN THE RECORD                    */
  2  FILIND      BIT(8),        /* FILE INDICATOR                                    */
                                /* 1XXXXXXX   THIS IS THE LAST VOLUME IN WHICH THE   */
                                /*            FILE RESIDES                           */
                                /* XXX1XXXX   CATALOGED FILE                         */
  2  ALLPARM,
                                /* ALLOCATION PARAMETERS                             */
  3  PARAM       BIT(8),        /* TYPE AND UNIT OF ALLOCATION:                      */
                                /* 00XXXXXX   ABSOLUTE ALLOCATION IN TRACKS          */
                                /* 01XXXXXX   IN BLOCKS                              */
                                /* 10XXXXXX   IN TRACKS                              */
                                /* 11XXXXXX   IN CYLINDERS                           */
                                /* XXXX1XXX   THE SPACE MUST BE CONTIGUOUS           */
                                /* XXXXXXX1   ROUND THE REQUESTED SIZE INTO A FULL   */
                                /*            NUMBER OF CYLINDERS                    */
  3  INCRSZ      BIT(24),       /* AUTOMATIC INCREMENT SIZE                          */
  2  LASTBLK,                   /* FOR SEQUENTIAL, PARTITIONED QUEUED                */
                                /* AND LINKED SEQUENTIAL:                            */
                                /*   RELATIVE BLOCK ADDRESS OF THE                   */
                                /*   LAST WRITTEN BLOCK (IN TTL, RL)                 */
```

```
                                /* FOR LINKED QUEUED FILES:                          */
                                /*   NUMBER OF AVAILABLE BLOCKS WITHIN THE FILE      */
                                /* FOR PARTITIONED QUEUED BEFORE FORMATTING (IN TTL): */
                                /*   NUMBER OF BLOCKS FOR THE DIRECTORY              */
    3  TTR,
    4  TTL          BIT(16),
    4  RL           BIT(8),
    3  LL           BIT(16),    /* FOR SEQUENTIAL AND PARTITIONED QUEUED:            */
                                /*   NUMBER OF BYTES REMAINING IN THE                */
                                /*   TRACK FOLLOWING THE LAST BLOCK                  */
                                /* FOR A LINKED SEQUENTIAL FILE:                     */
                                /*   FILE IDENTIFICATION NUMBER                      */
                                /* FOR A LINKED QUEUED FILE:                         */
                                /*   CURRENT SUBFILE IDENTIFICATION NUMBER           */
    2  RS3          CHAR(2),    /* NOT USED                                          */
    2  EXTDES (3),
                                /* DESCRIPTION OF THREE EXTENTS                      */
    3  EXTYPE       BIT(8),     /* TYPE OF THE EXTENT:                               */
                                /* 00000000  THIS FIELD INDICATES                    */
                                /*           THIS IS NOT AN EXTENT                   */
                                /* 00000001  DATA AREA OR PRIME AREA                 */
                                /* 00000010  OVERFLOW AREA (RANDOM                   */
                                /*           AND INDEXED SEQUENTIAL)                 */
                                /* 00000100  INDEX AREA (INDEXED SEQUENTIAL)         */
                                /* 01000000  USER LABEL EXTENT                       */
                                /* 10000000  SPLIT CYLINDER                          */
                                /* 10000001  THE EXTENT CONSISTS OF FULL CYLINDERS   */
    3  EXTSNB       BIT(8),     /* EXTENT SEQUENCE NUMBER:                           */
                                /* THE FIRST EXTENT SEQUENCE NUMBER = 00 FOR A       */
                                /* MULTIVOLUME FILE, THE EXTENT SEQUENCE NUMBER      */
                                /* BEGINS AT 00 FOR THE FIRST EXTENT IN EACH VOLUME  */
    3  EXTSTART,                /* ADDRESS OF THE FIRST TRACK OF THE EXTENT (CCHH)   */
    4  CCS          BIT(16),
    4  HHS          BIT(16),
    3  EXTLAST,                 /* ADDRESS OF THE LAST TRACK OF THE EXTENT (CHHH)    */
    4  CCL          BIT(16),
    4  HHL          BIT(16),
    2  NEXTFL,                  /* ADDRESS OF THE NEXT LABEL FOR THE FILE (CCHHR)    */
                                /* FL2 FOR CATALOGS, RANDOM AND                      */
                                /*     INDEXED SEQUENTIAL FILES                      */
                                /* FL3 FOR THE FILES CONTAINING MORE                 */
                                /*     THAN THREE EXTENTS                            */
                                /* FOR A LINKED SEQUENTIAL FILE:                     */
                                /*     ADDRESS OF THE GLOBAL EXTENT LABEL            */
    3  CCN          BIT(16),
    3  HHN          BIT(16),
    3  RN           BIT(8);
```

4.2.8 H_DFL2

Function

Declares the structure for the file label 2. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="06"X or LABEL="07"X option.

Syntax

```
$H_DFL2
          [ORG={ALL | UFAS | LINKQD_V | LINKQD_F}]
          [,NAME={FL2 | l_identifier31}]
          [,PREFIX = l_identifier19]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

ORG — File organization for which the file label 2 structure is to be generated.

ALL: generates a general file label 2 structure. ALL is the default value.

UFAS: generates a file label 2 structure for all file organisations attached to the UFAS access method.

LINKQD_V: generates a file label 2 for a library file or a LINKQD file residing on a VBO disk volume.

LINKQD_F: generates a file label 2 structure for a library file or a LINKQD file residing on an FBO disk volume.

NAME — Name of the generated structure. Default value is FL2.

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value is "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DFL2 is as follows:

```
DCL  1  FL2,
     2  KEY02     BIT(8),      /* ALWAYS = "02"X                          */
     2  AM1       CHAR(43),
     2  KEYF2     BIT(8),      /* ALWAYS = "F2"X INDICATING A FILE LABEL 2 */
     2  AM2       CHAR(90),
     2  PTRFL3,                /* VBO DISK => LABEL 3 DISK ADDRESS         */
                               /* FBO DISK => NOT USED                     */
     3  CC3       BIT(16),
     3  HH3       BIT(16),
     3  R3        BIT(8);
```

Expansion with ORG=UFAS

The declaration generated by the macro expansion of $H_DFL2 PREFIX='UFAS_', NAME = FL2_UFAS, ORG = UFAS, is as follows:

```
DCL  1  FL2_UFAS ,
     2  UFAS_KEY02          BIT(8),        /* ALWAYS = "02"X                          */
     2  UFAS_STATUS,
     3  UFAS_FRMT           BIT(1),        /* ="1"B => THE FILE HAS BEEN FORMATTED    */
     3  *                   BIT(7),
     2  UFAS_FILE_VERSION   BIT(8),        /* ="00"X => UFAS NLS, ="01"X => UFAS LS   */
     2  *                   CHAR(1),
     2  UFAS_MAXSIZE        FIXED BIN(31), /* UFAS FILE MAXIMUM SIZE                  */
     2  *                   CHAR(36),
     2  UFAS_KEYF2          BIT(8),        /* ALWAYS = "F2"X INDICATING A FILE LABEL 2 */
     2  UFAS_L2FDK,                        /* THE FOLLOWING IS MEANINGFUL ONLY        */
                                           /* FOR A UFAS INDEXED FILE.                */
     3  UFAS_NUMKEY         FIXED BIN(15), /* NUMBER OF KEYS. FIRST IS PRIME KEY      */
     3  UFAS_L2FDKE (16),                  /* KEY LOCATION WITHIN THE RECORD */
     4  UFAS_KEYLOC         FIXED BIN(15), /* STARTING FROM BYTE 0                    */
     4  UFAS_KEYSIZE        LOGBIN(8),     /* SIZE OF THE CORRESPONDING KEY           */
     4  UFAS_TYPE,
     5  UFAS_DATATYPE       BIT(2),
     5  UFAS_DIR            BIT(1),
     5  UFAS_DUP            BIT(1),        /* ="1"B => DUPREC                         */
     5  *                   BIT(2),
     5  UFAS_CODE           BIT(2),
     2  *                   CHAR(24),
     2  UFAS_PTRFL3,                       /* VBO DISK => LABEL 3 DISK ADDRESS        */
                                           /* FBO DISK => NOT USED                    */
     3  UFAS_CC3            BIT(16),
     3  UFAS_HH3            BIT(16),
     3  UFAS_R3             BIT(8);
```

Expansion with ORG=LINKQD_V

The declaration generated by the macro expansion of $H_DFL2 PREFIX = 'LINKQD_VBO_', NAME = FL2_LINKQD_VBO, ORG = LINKQD_V, is as follows:

```
DCL  1  FL2_LINKQD_VBO,
     2  LINKQD_VBO_KEY02          BIT(8),   /* ALWAYS = "02"X                          */
     2  LINKQD_VBO_AMLDL201,
     3  LINKQD_VBO_AM1,
     4  LINKQD_VBO_FRMTTED        BIT(1),
     4  LINKQD_VBO_OVERFLOW       BIT(1),
     4  LINKQD_VBO_UNSTABLE       BIT(1),
     4  LINKQD_VBO_OV_CURID_FLG   BIT(1),
     4  LINKQD_VBO_CHG_SFID       BIT(1),
     4  LINKQD_VBO_AMLDL202       BIT(3),  /* NOT CURRENTLY USED                      */
     4  LINKQD_VBO_NBDIRBLK       FIXED BIN(15),
     4  LINKQD_VBO_DIR_OVAD,
     5  LINKQD_VBO_DIR_OVTT       LOGBIN(16),
     5  LINKQD_VBO_DIR_OVR        LOGBIN(8),
     4  LINKQD_VBO_CURID          LOGBIN(16),
     4  LINKQD_VBO_OV_INCR        LOGBIN(8),
     4  LINKQD_VBO_BAM_AD,
     5  LINKQD_VBO_BAM_TT         LOGBIN(16),
     5  LINKQD_VBO_BAM_R          LOGBIN(8),
     4  LINKQD_VBO_BEGDATA        FIXED BIN(15),
     4  LINKQD_VBO_MAXSIZE        LOGBIN(24),     /* LINKQD FILE MAXIMUM SIZE            */
     4  LINKQD_VBO_USERTYPE       CHAR(2),        /* LINKQD FILE TYPE => SL, CU, LM, SM, BI, */
                                                  /*                     OR BLANKS       */
     4  LINKQD_VBO_DIRTYPE,
     5  LINKQD_VBO_DIRTYPE_LINK   BIT(1),
     5  LINKQD_VBO_DIRTP          LOGBIN(7),
     4  LINKQD_VBO_BAMTYPE        LOGBIN(8),
     4  LINKQD_VBO_CRASH_COUNT    FIXED BIN(15),
     4  LINKQD_VBO_OV_CURID       LOGBIN(16),
     2  LINKQD_VBO_OLD_DVC,
     3  LINKQD_VBO_OLD_BEGDATA    FIXED BIN(15),
     3  LINKQD_VBO_OLD_NBLK_TRK   LOGBIN(8),
     2  LINKQD_VBO_AMLDL203       CHAR(15),       /* NOT CURRENTLY USED                  */
     2  LINKQD_VBO_KEYF2          BIT(8),         /* ALWAYS="F2"X INDICATING A FILE LABEL 2  */
     2  LINKQD_VBO_AM2            CHAR(90),
     2  LINKQD_VBO_PTRFL3,                        /* VBO DISK => LABEL 3 DISK ADDRESS    */
                                                  /* FBO DISK => NOT USED                */
     3  LINKQD_VBO_CC3            BIT(16),
     3  LINKQD_VBO_HH3            BIT(16),
     3  LINKQD_VBO_R3             BIT(8);
```

Expansion with ORG=LINKQD_F

The declaration generated by the macro expansion of $H_DFL2 PREFIX = 'LINKQD_FBO_', NAME = FL2_LINKQD_FBO, ORG = LINKQD_F, is as follows:

```
DCL  1 FL2_LINKQD_FBO,
     2 LINKQD_FBO_KEY02          BIT(8),          /* ALWAYS = "02"X                          */
     2 *,
     3 LINKQD_FBO_AM1,
     4 LINKQD_FBO_FRMTTED        BIT(1),
     4 LINKQD_FBO_OVERFLOW       BIT(1),
     4 LINKQD_FBO_UNSTABLE       BIT(1),
     4 LINKQD_FBO_MONOSUBFILE    BIT(1),          /* MONOSUBFILE LINKQD FILE FOR BFAS SEQ    */
                                                  /* ON FBO DISK                             */
     4 *                         BIT(4),
     4 LINKQD_FBO_NBDIRBLK       FIXED BIN(15),
     4 LINKQD_FBO_CURID          FIXED BIN(31),
     4 LINKQD_FBO_BAM_CI         FIXED BIN(31),
     4 LINKQD_FBO_MAXSIZE        FIXED BIN(31),   /* LINKQD FILE MAXIMUM SIZE                */
     4 LINKQD_FBO_BEGDATA        FIXED BIN(15),
     4 LINKQD_FBO_USERTYPE       CHAR(2),         /* LINKQD FILE TYPE => SL, CU, LM, SM, BI  */
                                                  /*                                 OR BLANKS  */
     4 LINKQD_FBO_DIRTYPE        LOGBIN(8),
     4 LINKQD_FBO_NBBAMBLK       LOGBIN(8),
     4 LINKQD_FBO_CRASH_COUNT    FIXED BIN(15),
     2 LINKQD_FBO_OVF,
     3 LINKQD_FBO_OVF_NB_BLK     FIXED BIN(31),
     3 LINKQD_FBO_OVF_NB_MAIN    FIXED BIN(31),
     3 LINKQD_FBO_OVF_NB_ALIAS   FIXED BIN(31),
     2 LINKQD_FBO_IN_BLKSIZE     LOGBIN(16),      /* GIVEN BLOCK SIZE                        */
     2 *                         CHAR(6),
     2 LINKQD_FBO_KEYF2          BIT(8),          /* ALWAYS = "F2"X INDICATING A FILE        */
                                                  /* LABEL 2                                 */
     2 LINKQD_FBO_AM2            CHAR(90),
     2 LINKQD_FBO_PTRFL3,                         /* VBO DISK => LABEL 3 DISK ADDRESS        */
                                                  /* FBO DISK => NOT USED                    */
     3 LINKQD_FBO_CC3            BIT(16),
     3 LINKQD_FBO_HH3            BIT(16),
     3 LINKQD_FBO_R3             BIT(8);
```

4.2.9 H_DFL3

Function

Declares the structure for the file label 3. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="06"X or LABEL="07"X option on VBO disks.

Syntax

```
$H_DFL3
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DFL3 is as follows:

```
DCL  1  FL3,
     2  KEY03 (4)    BIT(8),       /* KEY EQUAL TO (4)"03"X INDICATES A DISK FILE LABEL 3 */
     2  EXTDESK (4),               /* FOUR EXTENT DESCRIPTIONS, THEY ARE THE SAME FORMAT  */
                                   /* AS THE FIELD "EXTDES" IN DECLARATION OF THE DISK    */
                                   /* FILE LABEL 1                                        */
     3  EXTYPE2      BIT(8),
     3  EXTSNB2      BIT(8),
     3  EXTSTART2,
     4  CCS2         BIT(16),
     4  HHS2         BIT(16),
     3  EXTLAST2,
     4  CCL2         BIT(16),
     4  HHL2         BIT(16),
     2  KEYF3        BIT(8),       /* FIRST CHARACTER OF THE DATA FIELD, ALWAYS EQUAL TO  */
                                   /* "F3"X, INDICATES A DISK FILE LABEL 3                */
     2  EXTDESD (9),
                                   /* NINE MORE EXTENT DESCRIPTIONS SAME AS EXTDESK       */
     3  EXTYPE3      BIT(8),
     3  EXTSNB3      BIT(8),
     3  EXTSTART3,
     4  CCS3         BIT(16),
     4  HHS3         BIT(16),
     3  EXTLAST3,
     4  CCL3         BIT(16),
     4  HHL3         BIT(16),
     2  NEXTFL3,                   /* ADDRESS OF AN EVENTUAL NEW DISK FILE LABEL 3 (CCHHR)*/
                                   /* NOT SUPPORTED: MUST BE (5)"00"X                     */
     3  CC3          BIT(16),
     3  HH3          BIT(16),
     3  R3           BIT(8);
```

4.2.10 H_DFL5

Function

Declares the structure for the file label 5. This declarative provides a structure used to map the output of the primitive H_READVTOC used with the LABEL="18"X option on VBO disks.

Syntax

```
$H_DFL5
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

The declaration generated by the macro expansion of $H_DFL5 is as follows:

```
DCL  1  FL5,
     2  KEY05 (4)        BIT(8),      /* KEY EQUAL TO "05"X INDICATING IT IS A FILE LABEL 5 */
     2  AVAEXT (8),                   /* EIGHT AVAILABLE EXTENT DESCRIPTIONS                */
     3  TRK1             BIT(16),     /* RELATIVE TRACK ADDRESS OF THE                      */
                                      /* FIRST TRACK OF THE EXTENT                          */
     3  CYLNB            BIT(16),     /* NUMBER OF FULL CYLINDERS WITHIN THE EXTENT         */
     3  TRKNB            BIT(8),      /* NUMBER OF TRACKS IN THE EXTENT                     */
                                      /* BESIDES THE FULL CYLINDERS                         */
     2  KEYF5            BIT(8),      /* KEY EQUAL TO "F5"X INDICATING IT IS A FILE LABEL 5 */
     2  AVAEXTD (18),                 /* EIGHTEEN MORE AVAILABLE EXTENT DESCRIPTIONS        */
     3  TRK1D            BIT(16),     /* RELATIVE TRACK ADDRESS OF THE                      */
                                      /* FIRST TRACK OF THE EXTENT                          */
     3  CYLNBD           BIT(16),     /* NUMBER OF FULL CYLINDERS WITHIN THE EXTENT         */
     3  TRKNBD           BIT(8),      /* NUMBER OF TRACKS IN THE EXTENT                     */
                                      /* BESIDES THE FULL CYLINDERS                         */
     2  NEXTFL5,                      /* ADDRESS (CCHHR) OF THE NEXT FL5                    */
     3  CCN5             BIT(16),
     3  HHN5             BIT(16),
     3  RN5              BIT(8);
```

4.2.11 H_GETVOLINFO

Function

Gets information about a cartridge volume.

Syntax

```
$H_GETVOLINFO
              LABEL = i_char_6
              [,DEVICE_CLASS = i_char]
              ,INFO = o_structure ;
```

Parameters

LABEL	The external label name of the volume. For a library cartridge tape, it is the VOLSER of the external OCR/Bar Code label.
DEVICE_CLASS	The device class of the volume. Currently, only CT/LIB is supported. Default value is 'CT/LIB'.
INFO	The name of the output structure to receive the information about the volume. The structure must be defined using the H_DCVOLINFO primitive.

Normal return codes

DONE:	H_GETVOLINFO was successful.

Abnormal return codes

RSUNKN:	The volume specified has not been found.
DVCERR:	The device class specified is not supported (by H_GETVOLINFO).
MDNAV:	The volume mounting request has been rejected by the operator.
HARDMALF:	The function (H_GETVOLINFO) cannot be executed due to system malfunction.
DVOV:	The function (called in IOF mode) cannot be executed because there is no available device in the specified device class.

In addition to the above return codes, you may also get a return code from a primitive called by H_GETVOLINFO.

Comments

When the return code is DONE, the following rules determine the values set in o_structure:

	LABEL	VOLSER	STANDARD	NON STANDARD	WORK	WRITE PROTECT	INCOHERENT
1)	label	label	1	0	1/0	1/0	0
2)	label	label	0	1	0	1/0	0
3)	label	label	1	0	1/0	1/0	1

where:

- line 1 refers to a standard volume,
- line 2 refers to a non-standard volume, and
- line 3 refers to a case where VOLSER is standard but the external label is inconsistent with VOLSER.

4.2.12 H_MSINFO

Function

Returns information on a specified media.

Syntax

```
$H_MSINFO

          MEDIA = iv_structure_media
          ,INFO_STRUCT = o_structure
          [,MIRROR_STRUCT = o_structure];
```

Parameters

MEDIA	Pointer to structure declared by H_DCMEDIA. This structure must be filled by the user.
INFO_STRUCT	Name of the structure declared by H_DCMSINFO.
MIRROR_STRUCT	Name of the structure declared by H_DCMSINFO_MIRROR.

Normal return codes

DONE:	H_MSINFO successfully completed.

Abnormal return codes

ARGERR:	The H_DCMEDIA structure is not correctly initialized: (NBVOL <> 1, DVTYP <>"MS", etc.
WRONGDV:	Returned when the device is in HELD state.

NOTE:

The H_MSINFO primitive returns information on the specified media which may or may not be assigned concurrently.

If the MIRROR_STRUCT parameter is specified, the mirrored disk information is returned provided that the specified media has the mirrored attribute.

4.2.13 H_READVTOC

Function

Reads the Volume Table of Contents of a FBO or VBO disks.

The information read depends on the LABEL parameter and on the organization of the disk as follows:

- for VBO and FBO disks

  File LOGICAL LABEL and VOLUME LABEL.

- for FBO disks only

  File extents description, DFL02 and volume free extents description.

- for VBO disks only

  DFL01, DFL02, DFL03 and DFL05.

Syntax

```
$H_READVTOC
          i_structure_media
          ,LBLARRAY = o_structure
          ,NBENTRY = i_fb15
          ,LABEL = i_bit8 byte
          [, FROM = i_char44]
          [,TO = b_char44]
          [,OUTAREA = i_ptr];
```

Parameters

i_structure_media

Structure declared by the primitive H_DCMEDIA. Only the entry relative to the first volume is taken into account. This entry must be initialized by the user.

LBLARRAY

The array which receives the type and labels that are returned by the primitive. The format of the array is as follows:

```
DCL 1   LABEL_ARRAY   (NBENTRY),
    2   TYPE          CHAR(1),
    2   LABEL         CHAR(140);
```

NBENTRY

Number of entries in LBLARRAY array. The user has to provide a number of entries according to the following rules:

If LABEL = "01"X: 1 entry by file.

If LABEL = "06"X:
- 2 entries by file if the disk is FBO and the OUTAREA parameter may be specified. The maximum value for NBENTRY is 180, corresponding to 90 files. This value fits a 64K small segment for OUTAREA, or 1000, corresponding to 500 files. (This value fits a 300K large segment for OUTAREA. See "Using a Large Segment for OUTAREA", in this section).
- 4 entries by file if VBO.

If LABEL = "07"X:
- 2 entries if the disk is FBO and the OUTAREA parameter may be specified.
- 4 entries if VBO.

If LABEL = "08"X: 1 entry.

If LABEL = "18"X,
- 1 entry if the disk is FBO and the OUTAREA parameter must be specified.

– At least 2 entries if VBO: the first one for the volume label and the other(s) for H_DFL5(s). Refer to the ALMOST return code description.

LABEL

Specifies the type of labels to be returned by the primitive.

"01"X -> LOGICAL LABEL (for one or several files).
"06"X -> LOGICAL LABEL AND EXTENSIONS (for one or several files).
"07"X -> LOGICAL LABEL AND EXTENSIONS (for only one file).
"08"X -> VOLUME LABEL (for the disk).
"18"X -> VOLUME LABEL AND FREE EXTENTS (for the disk).

FROM

Specifies a character string which is to be used to select the file names residing on the disk volume concerned. It may be a precise External File Name (EFN) or a * or a *-expression.

FROM is mandatory when the LABEL parameter equals "01"X or "06"X or "07"X.

FROM is forbidden when the LABEL parameter equals "08"X or "18"X.

If FROM specifies an EFN then, depending on the LABEL parameter value, either the specified EFN is selected or all file names greater than or equal to the specified string are selected.

If FROM = * then all file names are selected.

If FROM = *-expression which is composed by a constant part whose length is greater than 1 and followed with the * character then, only file names matching the constant part of the given *-expression are selected. Note that FROM="*" is not a *-expression.

If a * character is found within an EFN or within the constant part of a *-expression, it will be

considered as part of the selected file names, so INVOICES.A*G0001 is a valid EFN and INVOICES.A*G* is a valid *-expression with INVOICES.A*G as constant part.

If FROM is given a generation file name then, it must be the absolute generation name of the file.

If LABEL = "01"X and if the TO parameter is not specified, it implies the description of the logical label either of one file (EFN) or of all the files on the disk (*).

If the TO parameter is specified it means either the head of the list of files whose names are greater than or equal to the specified EFN (EFN) or the head of the list of all the files (*). In the case where FROM specifies a *-expression, the given value must not be modified (i.e. must always contain the initial *-expression) and the TO parameter must always be specified (refer to the TO parameter and to the ALMOST return code described below).

If LABEL = "06"X and if the TO parameter is not specified, the FROM parameter must be "*". It implies the description of the logical label and extensions of all the files residing on the disk (*).

If the TO parameter is specified it means either the head of the list of files whose names are greater than or equal to the specified EFN (EFN) or the head of the list of all the files (*). In the case where FROM specifies a *-expression, the given value must not be modified (i.e. must always contain the initial *-expression) and the TO parameter must always be specified (refer to the TO parameter and to the ALMOST return code described below).

If LABEL = "07"X, only a precise External File Name may be entered.

TO	Specifies a character string which is to be used either as input parameter or as output parameter by H_READVTOC. TO is meaningful when the LABEL parameter equals "01"X or "06"X and must not be used with the other LABEL values. It should be specified when FROM equals a *-expression. As input parameter TO is used to limit the selection of file names. It must be a precise External File Name (EFN) or a * or filled with blank characters. It must not be a *-expression. It means the end of a list of files whose names are smaller than or equal to either the specified EFN or any file ("*"). If FROM equals a *-expression, TO specifies the starting EFN within the file names list deduced from the *-expression. If LABEL = "06"X and if the TO parameter is filled, it must be greater than the FROM parameter. It must be filled if the FROM parameter is an EFN. If TO is given a generation file name then, it must be the absolute generation name of the file. As output parameter, it is used if the array table is too small. The returned value can be used either as the next FROM parameter (case where FROM equals a precise EFN or *) or must not be modified (case where FROM equals a *-convention) when the H_READVTOC primitive is called again.
OUTAREA	This parameter specifies the pointer to the segment created by the user to store the descriptions of the extensions in case of FBO disk. If LABEL = "07"X or "18"X, the segment created must be an expandable segment with an initial size of 8K Bytes. This segment may be expanded in the case where LABEL="18"X is used and the

concerned disk volume has a great number of free extents
(more than 370).

If LABEL = "06"X, the segment created must have an initial size conforming to the following formula:

```
SIZE >= (((NBENTRY / 2) - 1) * 600) + 8192
```

OUTAREA is mandatory when LABEL="18"X and optional when LABEL="06"X or "07"X. In the latter case, if OUTAREA is not specified or given a NULL() value then, the extents descriptions will not be returned.

Normal return codes

ALMOST: Returned when the LABEL parameter equals "01"X or "06"X and when FROM and TO parameters describe a list of files. The LBLARRAY array is too small to store the values returned by H_READVTOC so, the primitive is to be called once again with either the parameter FROM set to the value of the parameter TO as returned by the previous call (case where FROM is a precise EFN or *) or with the FROM and TO parameters not modified (case where FROM equals a *-expression).

This return code is also returned when H_READVTOC is used with LABEL="18"X on a VBO disk and the LBLARRAY array is too small to receive all the disk FL5's that describe the disk free space. In order to get all the FL5's, the primitive is to be called once again with a larger value for the NBENTRY parameter and a corresponding larger LBLARRAY array.

DONE: H_READVTOC has successfully completed. Note that the disk may be empty.

LIMIT:	Returned when LABEL="01"X or LABEL="06"X. It specifies that the H_READVTOC limit of 900 supported files is reached for a VBO disk. Only the 900 first files encountered in the VTOC are returned to the caller, the others are ignored.

Abnormal return codes

FLABUNKN:	Specified file is not found on the disk.
ARGERR:	Error in the given parameters. Check all parameters and especially LABEL value, FROM value, TO value, LBLARRAY size, NBENTRY value, OUTAREA size. Note that when LABEL="06"X is used NBENTRY must be a multiple of 2 if a FBO disk is concerned and a multiple of 4 if a VBO disk is concerned.
ABNORMAL:	Internal error.
DVIDFBID:	Conflict between the device attributes, device type and media name in the H_DCMEDIA structure.

Some other return codes not documented above may also be returned to the caller by H_READVTOC especially by the Disk Storage Manager (DSMGT).

Comments

The H_READVTOC primitive is used to read labels from the Volume Table Of Contents (VTOC) of a disk volume. FBO and VBO disks are supported (refer to the H_MSINFO primitive described in this manual for the difference between a FBO disk and a VBO disk). H_READVTOC can be used in multi-process context.

The NBENTRY parameter is used to provide a number of entries to the LBLARRAY array. The user is warned that giving a NBENTRY value larger than the number of LBLARRAY array entries may lead to unpredictable results such as system exception or spoilt data. When LABEL = "06"X, the user is warned that giving a NBENTRY value smaller than 50 may decrease performance significantly.

The OUTAREA parameter may be used with FBO disk volume. It provides a segment that is to contain either the file extents descriptions (LABEL="06"X or "07"X) or the disk volume free extents descriptions (LABEL="18"X). In the latter case, the user is warned that giving (via OUTAREA) a pointer pointing to a stack frame area may lead to the same unpredictable results as mentioned above. When LABEL = "06"X, the segment pointed by OUTAREA contains the file extents descriptions of each file. The size given for each description is 600 bytes ordered in the same way as in the LBLARRAY array.

The parameters FROM and TO are to be used in the following way.

LABEL = "01"X:

To get the logical labels of one or a list of files.

For one file, only the FROM parameter must be filled. For a list (FROM equals a precise EFN or *), the FROM parameter is filled with the head of the list and the TO parameter with the end of the list. One or both of these parameters can be "*". If FROM equals a *-convention, TO must be specified and equals either to blank characters or to an EFN.

If the array the user furnished is too small to receive the entire list described via the FROM and TO parameters, then the TO parameter will be filled, in output, by either the value of the next file to be described (case where FROM equals a precise EFN or *) or with the starting EFN within the file names list deduced from the *-expression. Then, the user has to call again the primitive with either the FROM parameter initialized with the last output value of the TO parameter or with the FROM and TO parameter not modified.

LABEL = "06"X.

To get the logical labels and extensions of one or a list of files.

Refer to LABEL="01"X as described above. TO is mandatory except when FROM is "*", and TO must be greater than or equal to FROM.

LABEL = "07"X.

Only one file name must be furnished. So only FROM must be filled by the user. TO must be initialized to BLANK.

LABEL = "08"X or "18"X

The information furnished by the primitive covers the volume, so there is no need of a filename. FROM & TO must be initialized to BLANK.

Support of Files Fragmented in more than 16 Extents

Usually the space allocated to a file or a part of a file residing on a disk volume cannot be fragmented in more than 16 extents. Starting from the TS 7458 Technical Status of GCOS 7-V7, a configuration option pushes the 16-extents limit to 255 for FBO files.

The values "06"X and "07"X for the label parameter allow the user to get file extents either for a set of files or for a single file.

When LABEL = "06"X is used, extents for each file are returned within an array specified via the OUTAREA parameter. Each entry of this array may be mapped by the H_DCEXTDESC primitive and is 600 bytes in length, and so can only contain 16 extents.

To support files fragmented in more than 16 extents with the LABEL = "06"X option the following is done:

the NBNEXT_LB16 field of the H_DCEXTDESC structure is duplicated into the REAL_NBNEXT_LB16 field of the same structure.

if the file has more than 16 extents the NBNEXT_LB16 field is reduced to 16.

So, for such files the user has to issue new H_READVTOC calls with LABEL = "07"X in order to get all the extents' descriptions.

Using a large segment for OUTAREA

When LABEL="06"X is used for an FBO disk in order to retrieve the extents descriptions for a very large number of files, specify a value of 1000 for NBENTRY and a 300K large segment for OUTAREA. This increases performance, and is done as follows:

- for the LBLARRAY array either create a large STACK3 segment by using the STACK3=(MAXSIZE=160K) command at linking time of the load-module, or use the $H_CRSEG GPL primitive to create a 160K large segment
- for the OUTAREA array use the $H_CRSEG GPL primitive to create a 300K large segment
- for large segment creations, reserve, at load-module linking time, the necessary two vacant entries in STN1, 2 or 3, e.g.:

```
SEGTAB3 = (SHRLEVEL=3, VSEG=2)
```

4.2.13.1 Summary of the Input Parameters

LABEL	FROM	TO	NBENTRY min value	OUTAREA	
1	EFN or *	EFN or *	1	-	VBO
					FBO
6	EFN or *	EFN or *	4 / file	-	VBO
			2 / file	user ptr	FBO
7	EFN	-	4	-	VBO
			2	user ptr	FBO
8	-	-	1	-	VBO
					FBO
18	-	-	2 (*)	-	VBO
			1	user ptr	FBO

(*) refer to LABEL="18"X detailed description above.

4.2.13.2 Description of the LBLARRAY Output Data

LABEL = "01"X (VBO or FBO)

Same visibility in VBO and FBO.

Entry	TYPE = "L" LABEL may be mapped by the declarative H_DCLOGLAB.

LABEL = "06"X (VBO case)

The array has four entries per file.

Entry i	TYPE = "L" LABEL may be mapped by the declarative H_DCLOGLAB.
Entry i+1	TYPE = "1" LABEL may be mapped by the declarative H_DFL1.
Entry i+2	TYPE = "2" LABEL may be mapped either by the declarative H_DFL2 or by 140("00"H) if the corresponding file has no DFL02.
Entry i+3	TYPE = "3" LABEL may be mapped either by the declarative H_DFL3 or by 140("00"H) if the corresponding file has no DFL03. Note that a file has a DFL03 onto the volume specified via H_DCMEDIA if its occupied space on this volume is fragmented into more than 3 extents.

LABEL = "06"X (FBO case)

The array has two entries per file.

Entry i	TYPE = "L" LABEL may be mapped by the declarative H_DCLOGLAB.
Entry i+1	TYPE = "A" LABEL may be mapped either by the declarative H_DFL2 or by 140(" ") if the corresponding file has no DFL02.

A segment may be specified by the user to put the corresponding file extents descriptions into it. It is pointed by the OUTAREA parameter.

Extent	For file i, the extents description is located at 600(i-1) bytes from the beginning of the segment. Each description contained in a 600 bytes area may be mapped by the declarative H_DCEXTDESC.

LABEL = "07"X (VBO case)

The array has four entries.

Entry 1	TYPE = "L" LABEL may be mapped by the declarative H_DCLOGLAB.
Entry 2	TYPE = "1" LABEL may be mapped by the declarative H_DFL1.
Entry 3	TYPE = "2" LABEL may be mapped either by the declarative H_DFL2 or by 140("00"H) if the file has no DFL02.
Entry 4	TYPE = "3" LABEL may be mapped either by the declarative H_DFL3 or by 140("00"H) if the corresponding file has no DFL03. Note that a file has a DFL03 onto

the volume specified via H_DCMEDIA if its occupied space on this volume is fragmented into more than 3 extents.

LABEL = "07"X (FBO case)

The array has two entries.

Entry 1	TYPE = "L" LABEL may be mapped by the declarative H_DCLOGLAB.
Entry 2	TYPE = "A" LABEL may be mapped either by the declarative H_DFL2 or by 140(" ") if the corresponding file has no DFL02.

A segment may be specified by the user to put the file extents description into it. It is pointed by the OUTAREA parameter.

Extent	The segment may be mapped by the declarative H_DCEXTDESC.

LABEL = "08"X (VBO and FBO)

The array has only one entry.

Entry	TYPE = "V" LABEL may be mapped by the declarative H_DCEXTVOL

LABEL = "18"X (VBO case)

The array has at least two entries.

Entry 1	TYPE = "V" LABEL may be mapped by the declarative H_DCEXTVOL
Entry 2	TYPE = "5" LABEL may be mapped by the declarative H_DFL5.

Other entries	TYPE = "5" LABEL may be mapped by the declarative H_DFL5.

LABEL = "18"X (FBO case)

The array has one entry.

Entry	TYPE = "V" LABEL may be mapped by the declarative H_DCEXTVOL.

A segment must be created by the user to put the free extents description into it. It is pointed by the OUTAREA parameter.

Extent	The segment may be mapped by the declarative H_DCEXTDESC.

Full expansion of the above mentioned primitives are given in this document.

Note that all the unused entries of the LBLARRAY array are filled with the character "FF"X up to the NBENTRY value regardless of the LABEL parameter value.

4.3 Device-level Primitives

4.3.1 H_DCDEVINFO

Function

Declares the structure DEVINFO referenced as output by H_DEVINFO primitive.

Syntax

```
$H_DCDEVINFO
          [,PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1];
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value "AUTOMATIC".
NLVL1	Used to hide the level one sub-structure name of the declarative structure.

Expansion

```
DCL  1  DEVINFO,
     2  VSN                CHAR (6),
     2  DEV_READY          BIT (1),
     2  ACTION_INPG        BIT (1),
     2  DEV_HELD           BIT (1),
     2  DEV_ALLOCATED      BIT (1),
     2  VOL_PROTECT        BIT (1),
     2  VOL_REQUESTED      BIT (1),
     2  VOL_LABEL_TYPE     BIT (1),
     2  VOL_WORK           BIT (1),
     2  NV_REQUESTED       BIT (1),
     2  DEVINFO_RFU        BIT (31);
```

Field Descriptions

VSN	name of the volume (Volume Serial Number).
DEV_READY	"1"B –> the device is ready, "0"B –> the device is on standby.
ACTION_INPG	"1"B –> action in progress. Action may be either Automatic Volume Recognition, or Rewind, or Unload.
DEV_HELD	"1"B –> the device is held (result of a previous MDHW xx OUT command issued by an operator or by the system). WHEN DEV_HELD IS SET TO "1"B, ALL THE OTHER INFORMATION IS NOT SIGNIFICANT.
DEV_ALLOCATED	"1"B –> the device is allocated to a job.
VOL_PROTECT	"1"B –> volume is physically protected.
VOL_REQUESTED	"1"B –> a volume mounting request has been issued for this device.
VOL_LABEL_TYPE	"1"B –> native label, "0"B –> non-standard label.
VOL_WORK	"1"B –> mounted volume is WORK.

NV_REQUESTED "1"B –> NV command (name volume) requested.

DEVINFO_RFU reserved for future use.

The possible combinations are summarized in Table 4-1 below.

Table 4-1. Device states and Volume Information

DEV_ HELD	DEV_ READY	ACTION_ INPG	DEV_ ALLOC	VOL_ REQUESTED	VOLUME INFORMATION
1	N/A	N/A	N/A	N/A	N/A
0	0	0	0	0	N/A (no volume mounted or requested)
0	0	0	1	1	REQUESTED VOLUME (volume mounting requested on standby device)
0	1	0/1	0	0	PREMOUNTED VOLUME
0	1	0/1	1	0	ALLOCATED VOLUME (mounted volume used by a job)
0	1	0/1	1	1	REQUESTED VOLUME (volume mounting requested on a device where a volume is already mounted)

ACTION_INPG = "1"B or VOL_REQUESTED ="1"B are transitory states. When one of these bits is found set, you are recommended to reissue the primitive a moment later.

If the volume information concerns a requested volume (VOL_REQUESTED="1"B), the work indication is found in the VSN field (it contains the string "WORK") and the bit VOL_WORK is not significant.

4.3.2 H_DEVINFO

Function

Returns information about a specified device.

Syntax

```
$H_DEVINFO
          DEVICE = iv_char4
          ,INFO_STRUCT = o_structure ;
```

Parameters

DEVICE — External Device Name.
The device type is 2 characters, and the device name is 2 characters. The allowed device types are: MS, MT and CT.

This parameter may be declared as follows:

```
DCL 1  I_STRUCT,
       2 DVTYPE   CHAR(2),  /* DEVICE TYPE */
       2 DVNAME   CHAR(2);  /* DEVICE NAME */
```

The I_STRUCT may be passed to the DEVICE parameter.

INFO_STRUCT — Name of the structure declared by H_DCDEVINFO.

Normal return codes

DONE: — H_DEVINFO has successfully completed.

Abnormal return codes

TPUNKN: — Unknown or illegal device type.

DVNAV: — Specified device type does not exist in the configuration.

Comments

The user is warned that applications using the H_DEVINFO primitive must be linked with the following LINKER command: NOLINK = H_DV_DVSTATE.

Also refer to the H_DCDEVINFO declarative.

4.3.3 H_DVLIST

Function

Lists the devices of a given type with the identification of the mounted volumes if any.

Syntax

```
$H_DVLIST
         device-type
         ,DVARRAY = o_list
         ,NBENTRY = i_fb15 ;
```

Parameters

device-type — i_char_2.
Device type, two characters: MS, MT, ...

DVARRAY — Array which receives the list of the devices of the given type with their characteristics.

```
DCL 1  LIST    (NBENTRY),
    2  DVTYP   CHAR(2), /*Device type       */
    2  DVATTR  BIT(16), /*Device attributes */
    2  DVNAME  CHAR(2), /*Device name       */
    2  VSN     CHAR(6); /*VSN = blanks if   */
                        /*there is no valid */
                        /*volume on this    */
                        /*device            */
```

The entries not used are initialized to blanks.

NBENTRY — Number of entries of the array reserved by the caller.

Normal return codes

DONE	H_DVLIST was successful.

Abnormal return codes

WAOV	Array overflow.
ARGERR	Device type unknown.

Comments

The different values of the DVATTR bit string according to the device types are not documented. The user may give DVTYP and DVATTR from the DVARRAY array as input to the H_INTDEV GPL primitive (refer to the *GPL System Primitives Reference Manual*) in order to translate the device class from the internal form to the GCL form.

There may be a difference between the DVATTR bit string returned by H_TRADEV and the DVATTR bit string filled by H_DVLIST. This is because H_TRADEV gives static information resulting from the device-class external form translation without accessing the device and H_DVLIST gives dynamic information got after device mounting. So, H_DVLIST may add to the static DVATTR returned by H_TRADEV some dynamic attribute bits. As an example; when a MS/FSA disk is concerned, H_TRADEV gives "8000"X and H_DVLIST gives "8080"X. The first DVATTR is included within the second and the logical intersection is true.

If no valid disk is mounted on the device whose name is given in DVNAME, the VSN character string is returned blank. To get more information about the device (HOLD, STANDBY, ...), you may use the H_DEVINFO primitive.

5. Automatic Operations SDI

The Automatic Operations SDI consists of the following primitives:

Access to the SYS.LOGC file:

H_CLOSELOGC	terminates SYS.LOGC processing
H_GETLOGC	obtains a record from the SYS.LOGC file
H_LOGCFD	defines the access method on the SYS.LOGC file
H_NOTELOGC	stores the current record address
H_OPENLOGC	starts SYS.LOGC processing
H_POINTLOGC	updates the SYS.LOGC record address

Job submission:

H_DCSPAWN	declares a structure for the H_SPAWNJOB primitive
H_SPAWNJOB	submits a job to the input reader

Elapsed time since last system restart:

H_DCRESTIME	declares the structure for the H_GTRESTIME primitive
H_GTRESTIME	obtains the time of the last system restart

The GPL syntax conventions are summarized in the preface of this manual. For a complete description of GPL syntax and of GPL in general, please refer to the GPL System Primitives Reference Manual.

The Automatic Operations SDI set of primitives has a specific MI (Marketing Identifier) which must be validated before program linkage.

5.1 SYS.LOGC Access

5.1.1 H_CLOSELOGC

Function

Terminates the processing on the SYS.LOGC File.

Syntax

```
$H_CLOSELOGC
            ifn ;
```

Parameters

ifn	Name of the LOGCFD structure.

Normal Return Codes

DONE

Abnormal Return Codes

IFNERR:	ifn is not a valid LOGCFD structure.
NOTOPEN:	File not open.

Comments

This primitive terminates the processing on the SYS.LOGC File. No other primitive can be issued for this file until a new H_OPENLOGC.

5.1.2 H_GETLOGC

Function

To get a record from the SYS.LOGC File.

Syntax

```
$H_GETLOGC
          ifn
          ,WA = o_location
          [,{BEGIN | INADDR = i_char6}]
          [,OUTADDR = o_char6]
          [,OUTLEN = o_fb15]
          [,ALN = i_fb15]
                                        {EDITED       }
          [,TIME = o_struc ,FORMAT = {ERLOG_BIN    }] ;
                                        {INTERNAL_BIN}
```

Parameters

ifn	Name of the LOGCFD structure.
WA	User area where the record will be moved.
BEGIN	Starts reading at begin of the file. The record got is the oldest of the file.
INADDR	Address of the record to be read.
OUTADDR	The specified location will receive the address of the current record.
OUTLEN	The specified location will receive the length of the record read.

ALN

This field contains the maximum record length expected by the caller. If a record exceeds that length, it is truncated and a WALIM return code is sent.

TIME

The specified output structure will receive the local logging-date of the read record, the format of which depends on the FORMAT parameter.

– if FORMAT = "EDITED":

```
DCL  o_struc CHAR(14);
     ==> contents: HHMMSSCCYYMMDD
                   EBCDIC encoded
```

– if FORMAT = "ERLOG_BIN":

```
DCL 1 o_struc,
      2 YEAR    BIT(16),
      2 DAY     BIT(16),
      2 TIME    CHAR(6);
     /* unit of time: 16ms. */
```

– if FORMAT = "INTERNAL_BIN":

```
DCL 1 o_struc,
      2 MILSEC   FIXED BIN(31),
      2 YEAR     FIXED BIN(15),
      2 DAY      FIXED BIN(15);
```

FORMAT

See TIME parameter above.

Normal Return codes

DONE

CHAINLIM:

A control record has been found. Next record will be of another CPU generation.

NORECFND:

The address given in INADDR (from H_GETLOGC or previous H_POINTLOGC) is valid but out of the file window.

TEMPLIM:

Temporary limit of the file has been reached. The caller has to wait for new introduction of new records before re-issuing a H_GETLOGC primitive.

WALIM:	Truncation occurred during move from buffer to WA (ALN < OUTLEN).

Abnormal Return Codes

ADDROUT:	Invalid address given in INADDR.
EXHAUST:	NORECFND already signaled.
FLNAV:	Physical file is not available.
IFNERR:	Ifn is not a valid LOGCFD structure.
INDERR:	Invalid address given in INADDR.
INDOUT:	Invalid address given in INADDR.
NOCURREC:	No current record available.
NOTOPEN:	File is not open.
UNRECIO:	Irrecoverable I/O error.

Comments

The relevant record will be moved from the access method buffer to the user Working Area with appropriate truncation if ALN parameter is underestimated. In addition, when HEADER = NO is specified (at LOGCFD level) the access method header (32 bytes long) is suppressed.

A SYS.LOGC record is mapped by the following structure:

```
DCL  1 log_record,
     2 log_header  CHAR(32)       /* record header  */,
     2 log_data    CHAR(n)        /* record data    */;
```

The desired record address may be:

- specified explicitly by the caller with INADDR option.
- implicitly specified with the BEGIN keyword and in this case, the first data record in chronological order will be returned.

After a successful H_POINTLOGC or H_GETLOGC operation, the caller can perform a H_GETLOGC without specifying BEGIN or INADDR clauses and in this situation, the address of the current record is be updated by the access method.

5.1.3 H_LOGCFD

Function

Declares the access characteristics on SYS.LOGC file.

Syntax

```
$H_LOGCFD
          ifn
          [,{ACTUAL | EMPTY}]
          [,ATTRIB = l_char]
          [,HEADER = {YES | NO}] ;
```

Parameters

ifn	Name by which the file is referred within the program.
ACTUAL	Allocation and initialization of the input table are done (default value).
EMPTY	No initialization is done and the HEADER parameter is ignored.
ATTRIB	Attributes to be given to the structure. When ACTUAL is specified, then a BASED attribute cannot be selected.
HEADER	Specifies whether the caller wants to receive the standard record header (built by the access method), or not. The default value is YES.

Comments

Declares and initializes the SYS.LOGC file structure used by the access method.

5.1.4 H_NOTELOGC

Function

Stores the current record address in user area.

Syntax

```
$H_NOTELOGC
          ifn
          ,OUTADDR = o_char6 ;
```

Parameters

ifn	Name of the LOGCFD structure.
OUTADDR	Location at which the internal form of the current record address is to be moved. (See parameter INADDR of H_POINTLOGC)

Normal Return codes

DONE

Abnormal Return Codes

FLNAV:	Physical file is not available.
IFNERR	Ifn is not a valid LOGCFD structure.
NOCURREC:	No current record available.
NOTOPEN:	File was not opened.

Comments

This primitive allows the caller to store the current record address at a given location (Specified by OUTADDR parameter).

5.1.5 H_OPENLOGC

Function

Initiates the processing on the SYS.LOGC File.

Syntax

```
$H_OPENLOGC
          ifn ;
```

Parameters

ifn	Name of the LOGCFD structure

Normal Return Codes

DONE

Abnormal Return Codes

FLNAV:	Physical file not available.
IFNERR:	Ifn is not a valid LOGCFD structure.
OPEN:	File was already opened.

Comments

This primitive initiates the processing on the SYS.LOGC File. It must be issued before any other primitive (except H_LOGCFD).

5.1.6 H_POINTLOGC

Function

Updates SYS.LOGC File record address for subsequent H_GETLOGC operations.

Syntax

```
$H_POINTLOGC
          ifn
          [,{BEGIN | INADDR = i_char6}] ;
```

Parameters

ifn	Name of the LOGCFD structure.
BEGIN	Makes the current record pointer pointed to the oldest record in chronological order.
INADDR	Address of the relevant record is given by the user.

Normal Return Codes

DONE

Abnormal Return Codes

ADDROUT:	Invalid address given in INADDR.
FLNAV:	Physical file is not available.
IFNERR:	Ifn is not a valid LOGCFD structure
INDOUT:	Invalid address given in INADDR.Ifn is not a valid LOGCFD structure.
NOTOPEN:	File was not opened.

Comments

This primitive must be issued before the first H_GETLOGC execution, exception provided for H_GETLOGC with BEGIN or INADDR parameter.

5.2 Job Submission

5.2.1 H_DCSPAWN

Function

Declares a structure which defines the processing to apply to the jobs spawned by H_SPAWNJOB primitive.

Syntax

```
$H_DCSPAWN
          [PREFIX = l_identifier]
          [,ATTRIB = l_char]
          [,NLVL1] ;
```

Parameters

PREFIX	Prefix value. Default value is no prefix.
ATTRIB	Storage class attribute. Default value is AUTOMATIC.
NLVL1	No level 1 generated.

Expansion

```
/*          $H_DCSPAWN;                              */
/*                                                   */
DCL 1  SPAWN_DESCRIPTION,
    2  SPAWN_LENGTH              FIXED BIN (15)
                                 INIT(46),            /* STRUCTURE LENGTH EXCLUDING VALUES*/
    2  SPAWN_CLASS               CHAR (2)
                                 INIT (" "),          /* JOB CLASS OF SPAWNED JOB         */
    2  SPAWN_PRIORITY            CHAR (1)
                                 INIT (" "),          /* JOB PRIORITY OF SPAWNED JOB      */
    2  SPAWN_HOLD                BIT (8)
                                 INIT ("00"X),        /* HOLD SPAWNED JOB                 */
    2  *                         BIT (8)
                                 INIT ("00"X),        /* RFU                              */
    2  SPAWN_USER,
    3  *                         CHAR (4)
                                 INIT ((4)" "),
    3  SPAWN_USERID              CHAR (12)
                                 INIT ((12) (" "),    /* TO MODIFY THE OWNER OF THE JOB   */
    2  SPAWN_JOB_SWITCHES,
    3  SPAWN_PASS                BIT (8)
                                 INIT ("00"X),        /* PASS SPAWNER JOB SWITCHES TO     */
                                                      /* SPAWNED JOB                      */
    3  SPAWN_SWITCHES            LOGBIN (32)
                                 INIT ("00000000"X),  /* PASS SWITCHES MASK TO SPAWNED JOB*/
    2  SPAWN_DELETE              BIT (8)
                                 INIT ("00"X),        /* DELETE SUBFILE                   */
    2  SPAWN_HOLDOUT             BIT (8)
                                 INIT ("00"X),        /* HOLD O/P CREATED BY SPAWNED JOB  */
    2  SPAWN_DESTINATION,                             /* OUTPUT DESTINATION               */
    3  SPAWN_PRIMARY_STATION     CHAR (8)
                                 INIT ((8) " "),      /* PRIMARY STATION NAME             */
    3  SPAWN_SECONDARY_STATION   CHAR (8)
                                 INIT ((8)" "),       /* SECONDARY STATION NAME           */
    2  SPAWN_VALUES_DESCRIPTION,                      /* PASSED VALUES TO SPAWNED JOB     */
    3  SPAWN_VALUES_LENGTH       FIXED BIN (15)
                                 INIT (0),            /* LENGTH OF THE STRING             */
    3  SPAWN_VALUES_STRING       CHAR (3000);
/*****************************************************************************************/
/*                                  END H_DCSPAWN                                        */
/*****************************************************************************************/
```

Field Description

SPAWN_CLASS

The job is attached to the job class defined by a letter. There are 16 job classes available from A up to P. The job class concept allows the definition of default scheduling and execution priorities and a maximum multiprogramming-level for jobs of the class. Also a class can be suspended and reactivated later upon operator's decisions. The use of job classes allows to control the job selection processing and manage serial execution of jobs.Class usage may be restricted on a PROJECT identification basis (catalog).

Default value:

- If class usage is controlled by project, then the default value is the default value attached to the PROJECT name.
- If the catalog does not exist or if there is no class usage restriction, then the default is the system default value P.
- If class usage restrictions do not fit user class specification, then a WARNING message is issued and the PROJECT default value is forced.

SPAWN_PRIORITY

Scheduling priority. Jobs are scheduled according to their scheduling priority (FIFO in case of same priority). 0 is the highest prior and 7 the lowest. A high priority job may not be scheduled if the maximum number of jobs of the same class are already scheduled. The default value depends on the job class. See the CLASS parameter.

Scheduling priority may be controlled on a PROJECT name basis if a SITE catalogue exists and specifies scheduling priority control. If the user specified priority exceeds the maximum allowed for the project then the highest value allowed for the project is forced and a WARNING message is issued.

SPAWN_HOLD	If this is 01, the job is created in the HOLD status after translation and if no error occurs. It is therefore not scheduled for execution until an operator types the RJ RON (RELEASE_JOB) command. Default value is NO HOLDEP STATUS: SPAWN_HOLD = "00"X;
SPAWN_JOB_SWITCHES	A set of 32 switches is associated with each executing job; (SW 0 to SW 31). The programmer can test the switches by the H_TESTSW primitive or the JUMP JCL statement. The switches allow the job flow to be controlled.
SPAWN_PASS	If is 01, the initial switches for the spawned job are those of the launching job as they are set at H_SPAWNJOB execution time. Default value is "00"X. The switches are not passed.
SPAWN_SWITCHES	Initializes the switch values being passed to the spawned job. Each bit set to 1 in the mask initializes the corresponding switch to 1. The leftmost bit corresponds to SW0 and the rightmost bit to SW31. Example: to initialize SW1 and SW30 to 1, RUN-SWITCHES = "40000002"X. Default value is "00" (every switch to 0).
SPAWN_DELETE	If this is 01, the subfile which contains the JCL is deleted after a successful introduction by the Stream Reader. If it is 0 (default), the subfile is not deleted. This value is forbidden if the file is not a queued file.
SPAWN_HOLDOUT	If this 01, the output files created by the spawned job are not delivered automatically. They are held

and can be, for instance, processed by the SCANNER. To release outputs, an RO command (Release output) must be issued.

Default value is no held output SPAWN_HOLDOUT = "00"X;

SPAWN_PRIMARY_STATION

Name of the station to direct the output created by the spawned job. The station must be linked to the project of the user whose job is spawned (JOB JCL command of the spawned job).

See the Catalog Maintenance command CREATE_STATION (CRS) for the creation of a station. If this field is blank (default), the station is the submitter station.

SPAWN_SECONDARY_STATION

Name of a station which is linked to the primary station. Each output for this station is first analyzed by the primary station which dispatches them on the secondary station.

Default value is blank (no secondary station).

SPAWN_VALUES_LENGTH

Length of the values string. It must be less than 300.

SPAWN_VALUES_STRING

The format of this string is:

```
"([parameter_value_1[,parameter_value_2...]]
 [keyword1=keyword_parameter_value1,
 [keyword2=keyword_parameter_value2]])]"
```

It defines the actual value to use in place of any occurrence of &i (1<=i<=99) or &keyword j in the invoked JCL sequence. The values of parameter_value_i and/or keyword_parameter_value_j can be any string of up to 128 characters. If non-standard characters

are used, then the entire string must be enclosed in quotes. If the string i has less than 2 characters or if the string keyword j has less than 8 characters, one can avoid adding erroneous trailing characters by putting two vertical bars "||" between the parameter reference and the trailing characters.

5.2.2 H_SPAWNJOB

Function

Spawns a job from a sequential file

Syntax

```
$H_SPAWNJOB
          ifn
          ,RUNPTR = i_run_ptr
          ,RESULT = o_char4;
          [,SUBFILE = i_char31]
          [,JOB = o_info] ;
```

Parameters

ifn	i_char8 Internal file name of the file which contains JCL statement of the submitted job. It is declared by the H_FD ifn macro.
RUNPTR	Pointer into a structure mapped by the H_DCSPAWN macro (see H_DCSPAWN description). It gives the processing to perform for the submitted job (job class, hold,...).
RESULT	It is filled by the primitive. It gives the result of the submission of the job (the RON of the spawned job). This information can be used as an input for the H_JOBLIST primitive which gives the characteristics of the spawned job (see GPL System Primitives Reference Manual).

SUBFILE	Name of the subfile which contains the JCL statement (queued files).
JOB	It is filled by the primitive. It gives the DATE and TIME of the submission of the job when it is successful. DCL 1 JOB_INFO, 2 INT_DATE CHAR (5), 2 INT_TIME CHAR (6); DATE_FORMAT="YYDDD" where: YY is the year (e.g. 85) DDD is the number of days from the beginning of the year (e.g. 082) TIME_FORMAT="HHMMSS" where: H=hour, M=minute, S=second (e.g. 195344)

Normal return codes

DONE:	One job has been processed by the Stream Reader and successfully introduced (a ron has been given). NOTE: this does not mean that the job is scheduled. It can be aborted by the JCL translator in case of JCL errors.
TABOV:	More than one job would be submitted. More than 1 JOB JCL statement is founded. NOTE: the first job is submitted (the o-ron is valid for this job).

Abnormal return codes

SYSOVLD:	It is not possible to find a KJOB entry for this job (too many jobs in the system).
OPTERR:	May describe: – Error in H_DCSPAWN structure: the CLASS is specified and its value is not between A and P

or, the PRIORITY is specified and its value is not between 0 and 7.
– Error in description of the values structure.
– Error in processing the input file.

ASGERR: The file is a queued file and it is not assigned with the SUBFILE parameter (when SUBFILE is not specified).

In addition see H_OPEN, H_GET, H_CLOSE, H_RTFLASG return codes.

Comments

This primitive sends to the Stream Reader a set of JCL statements for spawning one job only (otherwise a TABOV return code occurs). These statements are in a sequential file (or subfile) specified by an IFN. At the end of the primitive execution, the RON of the spawned job is released back to the issuer.

The file must have the following characteristics:

format:	FILEFORM = BFAS or UFAS
organization:	FILEORG = SEQUENTIAL, LINKQD queued + REFMODE = LOCATE

The primitive opens the file in input mode (PMD = INPUT) and processes sequentially the file (ACCMODE = SEQUENCE).Then, the file must be assigned before calling the primitive by $ASSIGN JCL statement or $H_ASSIGN GPL primitive.

In the case of a subfile of a queued file: SHARE must be DIR

If SUBFILE parameter is not specified, the file must be assigned with SUBFILE option. If specified, only the File needs to be assigned (no SUBFILE option).

For a sequential file: SHARE must be NORMAL

This primitive reads JCL statements from the input file and checks the presence of JOB, ENDJOB and allocates a RON number to the job.

- writes JCL statements in a system file
- writes input enclosure (if any) - INPUT, ENDINPUT - in the SYS.IN file.

- notifies the translator to process the JCL statements:
- closes the input file
- gives back control to the primitive issuer.

No action is taken on the file. It is the responsibility of the caller to manage its file (deassign, delete,...)

The caller specifies the processing to perform on the spawned job in the structure pointed by i_run_ptr (RUNPTR parameter) of the H_DCSPAWN primitive.

The H_SPAWNJOB primitive may not be used in a multi-process process group, except in the main process.

5.3 Elapsed Time since System Restart

5.3.1 H_DCRESTIME

Function

Declares the output structure for the elapsed time since the last system restart returned by the H_GTRESTIME primitive.

Syntax

```
$H_DCRESTIME

          [PREFIX = l_identifier]

          [,ATTRIB = l_char]

          [,NLVL1] ;
```

Parameters

PREFIX	Prefix of the declarative structure variables.
ATTRIB	Used to qualify the declarative structure.
NLVL1	Used to hide the level 1 sub-structure name of the declarative structure.

Expansion

```
/*__**__    $H_DCRESTIME;    */

DCL  1  RESTART_TIME,
     2  YEAR           FIXED BIN(15),
     2  DAY            FIXED BIN(15),
     2  TIME           FIXED BIN(31);
```

5.3.2 H_GTRESTIME

Function

To get the time of the last restart of the system.

Syntax

```
$H_GTRESTIME

          TIME_PTR = i_ptr ;
```

Parameters

TIME_PTR	Pointer to the H_DCRESTIME structure.

Return codes

None

Index

I

Technical publication remarks form

Title:	SDI-GPL Primitives Reference Manual

Reference No.:	47 A2 65UL 06

Date:	December 2006

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please include your complete mailing address below.

NAME: _____ Date: _____
COMPANY: _____
ADDRESS: _____
_____

Please give this technical publication remarks form to your BULL representative or mail to:

Bull - Documentation Dept.
1 Rue de Provence
BP 208
38432 ECHIROLLES CEDEX
FRANCE
info@frec.bull.fr

Technical publications ordering form

To order additional publications, please fill in a copy of this form and send it via mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone: +33 (0) 2 41 73 72 66
FAX: +33 (0) 2 41 73 70 66
E-Mail: srv.Dupilcopy@bull.net

CEDOC Reference #	Designation	Qty
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
[_ _] : The latest revision will be provided if no revision number is given.		

NAME: ______________________________ Date: ____________

COMPANY: ______________________________

ADDRESS: ______________________________

____

PHONE: ____________________ FAX: ____________________

E-MAIL: ______________________________

For Bull Subsidiaries:
Identification: ______________________________

For Bull Affiliated Customers:
Customer Code: ______________________________

For Bull Internal Customers:
Budgetary Section: ______________________________

For Others: Please ask your Bull representative.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE No.
47 A2 65UL 06