DASL DICTIONARY

PROGRAMMING IN RMS

SYSTEMS

UPDATE IV

Date: 04Aug84

to Document: DASL DICTIONARY
Update: III
Date: 03Aug83

This update completely replaces previous versions of the DASL DICTIONARY.

Since the dictionary is still under construction, please send any corrections you might note to Gene Hughes via RMS Mail. Anyone, even with older versions may send in their name, location, and dictionary version, to get updates.

.....following is a summary of changes in this update.

Major DASL CHANGES

Most of the changes in this update reflect additions to the DASL include file definitions relevent to RMS 2.2.

Change indicator

All changes in this update are marked in the right hand column with a: |*.

Changes to FUNCTION SECTION

Add New FUNCTIONS: $TXWRITB and $UERMSG
Change FUNCTION result: $MAP4K result to D$CCODE

Changes in TYPES SECTION

```
$ABSHDR field $LIBSPID type changed to $NAMET
$SYSTINFO.$TMADJ is now [3] BYTE
$ACB.$ACBMULT field removed
$FCBAIMI initializer string
$PCRCIF2 type was SET now SETW
```

Changes in WORD SECTION

Added WORDS (defined values):

```
$ECSTM2, $ECSTM3, $ECTSK29, $ECWI001
$g.... (Workstation graphics characters:
       in new include file: D$WORKSTN)
$FFMTXFD $SKDKW20
$WS3FD4 (in type field $WSCONFDS.$WSCON3)
$WS2EFK0 (a flag in $WSCONF2)
$WSESC1, $WSK1CHR ($WSIO control words; plus a
                   lot of extended function
                   keyboard codes, and
                   Escape-Sequence-1 codes that
                   follow $WSESC1.)

                   See: FUNCTIONS $WSIO,
                   and LISTS $WS....
```

```
Change VALUES: $LIBMXPG from 59 to 57 (in $ABSHDR)
               $WSKCTLT from $WSKFULL to $WSKDLLR
               $WSIOFCL from $WSCURDF to $WSK1CHR
               $PRINORM from $PRIMAX/2 to $NRPRIOR/2

Change WORDS:  $PABF002 to $PABFDATA
               $SKWS7 to $SKWS823
```

PREFACE:

This dictionary is an alphabetically organized presentation of the ***DASL Language*** Interface to the ***RMS System Function Routines***.

The DASL Interface to RMS provided by Datapoint

26 DASL text *include* files, and
 3 Relocatable code libraries: D$LIB, FAR, and RMSUFRS

(The libraries define ***external***
functions, variables and values.)

The programmer will also require the DASL Compiler, the SNAP Assembler and the LINK and EASL Utilities to generate programs.

HOW THE DICTIONARY IS ORGANIZED

The **RMS** ***System Calls, File Access Routines,*** and ***User Function Routines*** are described in terms of DASL syntax and parameter requirements in the **FUNCTIONS SECTION**. ***DASL keywords*** are included in this section for convenient reference.

Function parameters which are of a ***predefined variable structure type*** may then be found described in the **TYPES and STRUCTURES SECTION**. (Other parameters which are standard DASL variable types will be fully described and commented with the Function.)

Pre-defined parameter values will be listed with the functions and types that use them, and their descriptions may be found in the **WORD SECTION** and in the **LIST SECTION** of flags and value groups. Also in the WORD section will be entries for all named functions, variables, structures, sub-structure fields, DASL keywords, and Datapoint abbreviations.

HOW TO USE THE DICTIONARY

Look CAPITALIZED words or abbreviations up in the **WORDS SECTION** ***if in doubt*** as to their type or meaning. Then look in either the TYPE or FUNCTION SECTIONS for additional description. See the information at the front of the WORDS section explaining the notation method of cross-referencing.

Also, the LISTS SECTION can be useful to find related functions, types and values.

HINTS:

1) ***Leading dollar signs*** "$" are not recognized in alphabetization. Example:

```
 D$GET24
 DBLBUFF$
$DCB
$DGETCRK
```

2) **Each section has format explanation at the beginning.**

NOTE:

This document provides referencing to the *RMS System Programmer's Reference Manual*

which should be available to the programmer for obtaining more specific details on some of the RMS system functions.

The descriptions here will give the DASL programmer sufficient orientation to the function syntax and names and to the parameter names, so that the assembly language descriptions will be useful.

The references to the SPRM will not be provided in subsequent editions of this dictionary, following this experimental limited edition.

The DATE of the SPRM referenced is SEPT 1982.

CONTENTS

LISTS

DASL OPERATOR SYMBOLS

This list is a combined description, order of precedence, and implied grouping (left-to-right or right-to-left) when several operators of the same precedence appear at the same level.

The groups are listed in order of precedence, highest first. Each group title will state the implied grouping. Operators in the same group have equal precedence. Precedence and grouping may both be modified with the use of parentheses.

See the DASL Document for further description and requirements for variable and value TYPES.

POST-UNARY (left-to-right):

```
^   Pointer, indirection. x CHAR:= 'A'; p ^CHAR:= &x;
               p^ := 'B'; is equiv. to x := 'B';
[]  Array Index
.   Field Operator; see TYPE $FCBAIM Program Example
()  Function parameter and DEFINE argument indicator,
    or evaluation grouping. (overides operator
    precedence)
++  Increment after evaluation
--  Decrement after evaluation
```

PRE-UNARY (right-to-left):

```
++  Increment before evaluation
--  Decrement before evaluation
&   "Address of" operator
-   Negative of operand, arithmetic
~~  One's complement of operand, arithmetic
~   Unary not; inverts TRUE / FALSE
SIZEOF See FUNCTION section: SIZEOF
<type> Cast operator; change type for this evaluation
```

BINARY group 1 (left-to-right):

*	Multiply
/	Divide
%	Modulo: (a%b), Remainder of a/b

BINARY group 2 (left-to-right):

+	Addition
-	Subtraction

BINARY group 3 (left-to-right):

<<	Shift Left; var := bits << numberOfBits
>>	Shift Right

BINARY group 4 (left-to-right):

&&	Bit-wise "and"

BINARY group 5 (left-to-right):

||	Bit-wise "inclusive or"
!!	Bit-wise "exclusive or"

BINARY group 6 (left-to-right):

Used in flow of control

=	Equal
~=	Not Equal
<	Less Than
>	Greater Than
<=	Less Than or Equal
>=	Greater Than or Equal

BINARY group 7 (left-to-right):

&	Logical "and"; expressions are true if not equal to zero.

BINARY group 8 (left-to-right):

|	Logical "or"

TERNARY (right-to-left):

? : Conditional; a<b ? trueExpressn : falseExpressn
Similar to IF THEN ELSE, example:

```
z := A <=  5 ? 5
   :  A <= 10 ? 10
   :  A <= 15 ? 15
   :            1;
```

BINARY group 9 (right-to-left):

Assignment Operators

:=	a := b; a gets b
*=	a *= b; a gets a*b
/=	a /= b; a gets a/b
%=	a %= b; a gets a%b
+=	a += b; a gets a+b
-=	a -= b; a gets a-b
<<=	a <<= b; a gets a<<b
>>=	a >>= b; a gets a>>b
&&=	a &&= b; a gets a&&b
||=	a ||= b; a gets a||b
!!=	a !!= b; a gets a!!b

BINARY group 10 (left-to-right):

,	Separator; parameters, values in strings or initializers.

SYMBOLS NOT SHOWN in DASL Document OPERATORS List:

;	Ends a statement. Note: DO NOT use following a macro call; like DEFINE (), INCLUDE () etc.
:	Label : Statement
{}	Statement grouping or initializer grouping

Used by DASL Compiler MACRO "pre-processor"

#	PARAMETER NUMBER; see DEFINE in FUNCTIONS
#[#]	EVALUATION SUPPRESSION; see DEFINE in FUNCTIONS

DASL WORDS AND FUNCTIONS

LISTS of RMS FUNCTIONS and TYPES follow.
DASL words are defined in the FUNCTION SECTION.

Functions DASL	DASL RESERVED WORDS
	DASL Compiler Defined
CASE	Transfer Control to Statement Label Equal to Argument
DEFAULT	Preceeds Statement in CASE for No-match Condition
DEFINE	Define a String to be Substituted for Identifier
ELSE	Part of IF THEN ELSE Execution Control
ENTRY	Declare Global Name; may be Ref Externally
EXTERN	Declare a Name Defined in Another Module
FAST	Future Code Generators; Var. Resides in Registers
GOTO	Transfer Control to Labled Statment in Same Function
IF	Execute THEN Statement if Expression is TRUE
IFELSE	If 1st 2 Strgs are Equal Reslt is 3rd,Else 4
INCLUDE	Obtain Program Input Lines from Specified File
INCR	Produce a Value by Incrementing the Argument by 1
LOOP	Execute Substatements Until WHILE Exprsn = 0
RECURSIVE	Specifies Function may be Called Recursively
RESULT	Assign Resultant Value to a Function
SIZEOF	Operator Which Gives Size in Bytes of Argument
STATIC	Prevents Re-allocation of Variable in Recursive Function
STRUCT	Named Member Consistng of Several Named Membrs
SUBSTR	Select Part of String, Begin at Start for Length Specified
SYSTEM	Reserved for Future Code Generators
THEN	Part of IF THEN ELSE Execution Control
TYPDEF	Give Type a name that may be used as a Type
UNION	Named Member Contains Different Possible Members
VAR	Indicates Local Variable Definitions Follow
WHILE	Part of LOOP WHILE Execution Control

DASL INCLUDE FILE DEFINED MACROS

D$INC Include File

Basic DASL definitions, Standard Include

ENUM	Define Var Type BYTE; Define Values 0 thru 8
ENUMV	Define Values Incrementing from Initial Value
SET	Define Var Type BYTE and Define Powers of 2
SETV	Define Ascending Powers of 2 from Initial Value
SETW	Define Var Type UNSIGNED; Define Powers of 2

DASL EXTERNALLY DEFINED FUNCTIONS

in D$LIB/REL Library File

D$INC Include File

Basic DASL definitions, Standard Include

D$CALL	Call an Arbitrary Externally Defined Subroutine
D$COMP	Block Compare Two Character Strings
D$GET24	Numeric, Convert 24 Bit to 32 Bit Value
D$INFO	Return Processor Type
D$MOVE	Block Move 0 to 65535 Bytes
D$MOVER	Block Move Reverse, Starting at Ending Address
D$PUT24	Numeric, Convert 32 Bit to 24 Bit Value
D$SC	Call a SYSTEM CALL External Function
RASLEND$	Turn off Traps in Program Running RASL
RASLRES$	Invoke the DASL Debugger

D$RMS Include File

Common Nucleus and UFR definitions

D$JUMP	Jump to an Arbitrary Externally Defined Subroutine

RMS FUNCTIONS

Functions are grouped first by System Calls, File Access Routines, and User Function Routines, and then by INCLUDE FILE.

Functions — System Calls — SYSTEM CALLS

D$RMS Include File

Common Nucleus and UFR definitions

$CLOSE	Close a File	
$ERROR	Abort a Program	
$EXIT	Exit a Program	
$LOAD	Load an Overlay	

D$RMSGEN Include File

RMS General System Function Definitions

$GETIME	Obtain Current System Time	
$INFO	Obtain System Configuration Information	
$SETIME	Set The Current System Time	
$SETSQL	Set Independent Task Security Level	|*

D$RMSIO Include File

File Handling, Block I/O, Disk, Printer,Pipe

$CLOSEAL	Close All Open Files	
$DISCONT	Disconnect from remote node	|*
$FILES	Multi-Resource, Obtain Disk File Information	
$FORMAT	Multi-Resource, Format a Unit on the Disk	
$GETSFI	Obtain Symbolic File Identification	
$OPENENV	Open a File with Specified ENV	
$PIPEGEN	Create a Pipe Resource	
$PIPEUSE	Check Local Pipe-in-use Status	
$RENENV	Change a Disk File Name	
$REOPEN	Reopen a File With New Passwords	
$SECCHK	Check Operation Status	
$SECEOF	Obtain or Set End Of File Location	
$SECR	Block Read	
$SECRO	Block Read Optimum	
$SECURE	Multi-Resource, Change Disk File Security	
$SECW	Block Write	
$SECWAIT	Wait for Operation Complete	
$SECWO	Block Write Optimum	
$STOPIO	Stop All Data Movement	
$WAITIO	Wait for any FAV Operation to Complete	|*
$WAITIOS	Wait for selected status bit to change	|*

D$RMSMEM Include File

Memory Management Definitions

$BASESET	Set the Memory Base Register
$MEMCTL	Memory Control
$MEMGET	Obtain PSK for Available User Mem. Sector
$MEMKEY	Obtain PSK of a Mapped Memory Sector
$MEMMAP	Map a Memory Sector into Logical Space
$MEMPROT	Change Memory Protection
$MEMREL	Release a Memory Sector
$SETMAX	Set Maximum Memory Requirement
$SETMIN	Set Minimum Memory Requirement

D$RMSPROG Include File

Program Loading and Execution Control

$GLUTEN	Get Last User Task Error Number
$RFIAKS	Return from Abort Key Seq Interrupt
$RFIDKS	Return from $TRAPDKS Interrupt
$RFIFK	Return from $TRAPFK Interrupt
$RFIKKS	Return from $TRAPKKS Interupt
$RFILKS	Return from$TRAPLKS Interupt
$RUN	Load and Run a Program
$SETPRI	Set User Task Priority Level
$SIGNON	Force User Signon
$TIMER	Reset System Timer
$TRAPAKS	Trap ABORT Key Sequence
$TRAPDKS	Trap DISPLAY-CANCEL-DISPLAY Key Sequence
$TRAPFK	Trap Function Key Strokes
$TRAPKKS	Trap KEYBOARD-CANCEL-KEYBOARD Key Sequence
$TRAPLKS	Trap LOG-OFF Key Sequence
$TRAPSET	Trap Setting System Call, Used Indirectly
$TRAPUMV	Trap User Mode Violations, Used Indirectly
$USRABN	User Abend Facility

D$RMSSPEC Include File

Special System Calls

$DONATFV	Donate a FAV to a Specified Task
$LOCKFAV	Lock / Unlock a Specified FAV
$RELFAVS	Release All Locked FAVs

D$RMSTASK Include File

User Multi-tasking

$SINDEP	Start an Independent Task
$SLOCAL	Start Local Task
$TASKCTL	Exert Control Over an Independent Task
$UCSCHK	Check a User Created Semaphore
$UCSDEL	Delete a User Created Semaphore
$UCSGEN	Generate a User Created Semaphore
$UCSSIG	Signal a User Created Semaphore
$UCSWAIT	Wait on a User Created Semaphore

D$RMSWS Include File

Workstation

$BEEP	Make a Beep Sound
$CLICK	Make a Click Sound
$CURSOFF	Turn Off the Cursor
$CURSON	Turn On the Cursor
$WCONFIG	Get Configuration
$WSCTL	Control Code Function,Used Indirectly
$WSGETCH	Obtain One Keyboard Buffer Character
$WSIO	Perform Workstation I/O
$WSTATUS	Get Status
$WSWAIT	Enable Datastation Port and Wait for Character

FILE ACCESS ROUTINES

D$FAR Include File

File Access Routines, REQUIRES D$RMS

$ACLOSE	Close an AIM File
$ADELCR	AIM Delete Curr Rec Read by $AREAD,$AREADKG
$AINS	Insert Key in Index for Existing Data Recd
$AIOCLR	Comp Pend Writes,Force Buffer Fill on Read
$AOPEN	Initialize AIM File Access
$APOS	Posit to Logical Rec Meeting Key List Spec
$AREAD	AIM Read Logical Rec Meeting Key List Spec
$AREADCR	Read after $APOS or Re-read Cur after $AREAD
$AREADKG	Read Key Generic, Read Records with Same Key
$ARWRTCR	Rewrite Cur Record after $AREAD, $AREADKG
$AWRITE	Write Recd at EOF Data File,Insert Key in Index
$DCLOSE	Direct Access, Close
$DDEL	Direct Random Access, Delete
$DDELCR	Direct Seq Access, Delete Current
$DGETCRK	Direct Seq Access, Get Current Record Key
$DGETNRK	Direct Seq Access, Get Next Record Key
$DIOCLR	Direct Access, I/O Clear
$DOPEN	Direct Access, Open
$DPOS	Direct Random Access, Position
$DPOSEOF	Direct Random Access, Position to EOF
$DPOSNX	Direct Sequential Access, Position to Next
$DPOSPV	Direct Seq Access, Position to Previous
$DREAD	Direct Random Access, Read
$DREADCR	Direct Sequential Access, Read Current
$DREADNX	Direct Sequential Access, Read Next
$DREADPV	Direct Sequential Access, Read Previous
$DRWRT	Direct Random Access, Rewrite
$DRWRTCR	Direct Sequential Access, Rewrite Current
$DWRITE	Direct Random Access, Write
$DWRITNX	Direct Sequential Access, Write Next
$DWRTEOF	Direct Sequential Access, Write End of File

...Continued

D$FAR Include File....Continued

$ICLOSE	ISAM, Close
$IDEL	ISAM Random, Delete
$IDELCR	ISAM Seq, Delete Current Record Key, Data
$IDELK	ISAM Random, Delete Record's Key
$IINS	ISAM Random, Insert
$IIOCLR	ISAM, I/O Clear
$IOPEN	ISAM, Open
$IPOS	ISAM Random, Position
$IPOSKP	ISAM Sequential, Position to Key Previous
$IPOSKS	ISAM Position to Next Key Sequential Record
$IPREP	ISAM, Prepare File
$IREAD	ISAM Random, Read
$IREADCR	ISAM Sequential, Read Current
$IREADKP	ISAM Sequential, Read Key Previous
$IREADKS	ISAM Sequential, Read Key Sequential
$IRWRT	ISAM Random, Rewrite
$IRWRTCR	ISAM Sequential, Rewrite Current
$IWRITE	ISAM Random, Write

USER FUNCTION ROUTINES

D$RMS Include File

Common Nucleus and UFR definitions

$ERMSG	Display RMS Minimum Error Message
$LBGTLSN	Locate Library Member and Return LSN
$MSGC	Err-Msg, Locate a Message and Copy Into $MSG
$OPEN	Open a File; Search for ENV, which is not Specified
$SCANFLS	CmdInt, Scan File Specs According to Table
$SCANOS	CmdInt, Scan Options Specification

D$UFRENV Include File

Environment Handling
Requires D$RMSIO include if $$FILENAM function used

$DECOHSI	Decompress an HSI String
$ENVDEL	Delete Existing Environment
$ENVFNDM	Find Environment Data Match
$ENVINS	Insert New Environment
$ENVLGET	Obtain Environment Entry Length
$ENVLOC	Locate Existing Environment
$ENVPDAT	Position to Environment Data in an Env Entry
$ENVPEEL	Prepare Open of Environment's Catalog file
$ENVPHSI	Position to HSI Name in an Env Entry
$ENVPLOP	Position to UET Link In Open Parameter Table
$ENVPNAM	Position to Env Name in an Env Entry
$ENVPNET	Position to Net Name in an Env Entry
$ENVPNOD	Position to Node Name in an Env Entry
$ENVPPAS	Position to First Password in Env Entry
$ENVPRES	Position to Resource Name in Env Entry
$$FILENAM	Obtain the Next File Name in Catalog
$$FILEPCN	Open Catalog File and Obtain PCNs
$$FILEPCU	Special Entry to $$FILEPCN
$GETPASS	Obtain, Compress Password from Keyin
$NEWPCR	Create New PCR for Independent Task
$PAKPW	Pack ACSII String Into Password
$SCANHSI	Compress HSI
$UNPAKPW	Unpack password into ASCII string

D$UFRERR Include File

Error Handling

$MSGCGET	Locate and Deliver a Message	
$MSGCXO	Open Command Library Message Members	|*
$RMSMSG	Return RMS message	|*
$UERMSG	Store User Error Message on Command Stack	|* |*

D$UFRGEN Include File

General Utility

$CVSTIME	Obtain System Date and Time Info
$DATETIM	Convert System Time to Standard
$DISORT	Diminishing Increment In-Core Sort,Ascendng
$FILEFMT	Convert File Format Codes to 4-Byte ASCII String
$GDATTIM	Obtain Current ASCII Date/Time String
$WIPEBT	Clear an Area of Memory to SPACES
$WIPEBTA	Clear an Area of Memory to Constant Value

D$UFRLIB Include File

Library Manipulation

$LBADD	Add a Member to a Library
$LBDEL	Delete a Member From a Library
$LBFIND	Locate Library Member
$LBFREE	Find the First Free Sector in a Library

D$UFRMEM Include File

Memory Management

$MMFREMEM	Deallocate a Block of Memory
$MMGETCLR	Allocate a Cleared Block of Memory
$MMGETFST	Allocate Free Space Block of Memory
$MMGETMEM	Allocate a Block of Memory
$MMGETPAG	Obtain a Page of Logical Memory Space
$MMINIT	Initialize Memory Management
$MMRETPAG	Release a Page of Logical Memory Space

D$UFRNQDQ Include File

NQ/DQ UFR Definitions

$NQDQBLD	Build a Message Block
$NQDQCHK	Check Limited Request
$NQDQENL	Request a Limited Enqueue
$NQDQENQ	Enqueue on a Resource
$NQDQLGF	Disconnect from the System
$NQDQLGN	Connect to the System
$NQDQREL	Release a Resource
$NQDQRST	Reset the Controller 4-byte Counters
$NQDQSTA	Acquire Controller Statistics
$NQDQSTP	Terminate an In-Progress Limited Enqueue
$NQDQWAT	Wait for a Limited Enqueue Request

D$UFRNUM Include File

Numeric Manipulation

$BINOC24	Convert 24-Bit Binary to ASCII Octal
$BINOCT	Convert 16-bit Binary to ASCII Octal
$CONDEC	Convert Decimal to Binary
$CONOC24	Convert ASCII Octal to 24-Bit Binary
$CONOCT	Convert ASCII Octal to 16-Bit Binary
$CVB	Convert ASCII Decimal to 16-Bit Binary
$CVB24	Convert ASCII Decimal to 24-Bit Binary
$CVB24L	Convert ASCII Decimal to 24-Bit Binary
$CVD	Convert 16-Bit Binary to ASCII Decimal
$CVD24	Convert 24-Bit Binary to Decimal
$CVD24Z	$CVD24 with Zero-Suppression
$CVDZ	$CVD with Zero-Suppression
$DIVID3	Unsigned 24-bit Division
$FDPACK	Pack Two Decimal Numbers
$FDUNPAK	Unpack Character Into Two ASCII Digits
$MLTPLY3	Unsigned 24-bit Multiplication
$SDIVID3	Signed 24-bit Division
$SMLPLY3	Signed 24-bit Multiplication

D$UFRRLD Include File

Relocating Loader

$LOADREL	Invoking the Relocating Loader

D$UFRSCAN Include File

Command Interpreter (Scanning)

$FSCAN	Compress a $FILESPK
$GENSMSK	Generate Generic Scanning Masks
$GENSTST	Name Test-Under-Mask and Generate
$SCANCFG	Read and Scan User Configuration File
$SCANFS	Scan a File Specification
$SCANNB	Scan to Next Non-Blank
$SCANSYM	Scan a Symbol
$SCANWRD	Scan Two Word Lists for Matches

D$UFRSYS Include File

System Interface

$LOCKRIM	RIM Lockout, Attempt to Open Pipe
$MAP4K	Allocate Memory For a PFDB
$$MEMGET	Obtain A Physical Memory Sector
$$MEMREL	Release a Physical Memory Sector
$UNLKRIM	Release RIM from Pipe ? (on hold)

D$UFRWFIO Include File

Work File I/O

$TAPEREWIND	Rewind Tape	
$TAPEUNLOAD	Rewind Tape and Unload	
$TXBKSP	Backspace a logical record	
$TXCLOSE	Terminate Processing for a Text-File	
$TXDEL	Delete a Logical Text Record	
$TXOPEN	Prepare an Opened Text-File for Access	
$TXOPENP	Open Using Specified Physical I/O Routine	
$TXPOSEF	Position to Text-File EOF	
$TXPOSIT	Position Text-File to File Pointer	
$TXPREP	Prepare a New Text-File for Access	
$TXPREPP	Prepare w/ Specific Phys. I/O Routine	|*
$TXREAD	Read a Logical Text-File Record	
$TXUPDATE	Update a Logical Text Record	
$TXWEOF	Write EOF at Current Text-File Position	
$TXWRITB	Write Text-File Record of Specified Length	|* |*
$TXWRITE	Write a Logical Text-File Record	
$VGETBUF	Obtain Buffer Group from Virtual Pool	
$VINIT	Initialize Virtual I/O Management	
$VMAPPSK	Donate a PSK to Virtual Management	
$VPUTBUF	Return a Buffer Group to Virtual Pool	
$VSETWIN	Establish Memory Window Areas, Virtual	
$WFCLOSE	Terminate Processing of Work File	
$WFFLUSH	Dump Pending Write Buffers to Disk	
$WFOPEN	Prepare an Open Work File For Access	
$WFOPENP	Open Using Specified Physical I/O Routine	
$WFPOSEF	Position to File's EOF	
$WFPOSIT	Position to File Pointer	
$WFPREP	Prepare a New Work File For Access	
$WFPREPP	Prepare Using Specified Physical I/O Routine	
$WFREAD	Read a Logical Record	
$WFREADL	Read in LOCATE Mode	
$WFUPDATE	Update a Logical Record	
$WFUPDATEL	Update a Record in LOCATE Mode	
$WFWEOF	Write EOF At Current File Position	
$WFWRITE	Write a Logical Record	

$WFWRITEL	Write a Record in LOCATE Mode

Functions
UFR

D$UFRWS Include File

Workstation Interface

$CHAININ	Determine if CHAINing is Active
$$CLOSEAL	Interface to $CLOSEAL System Call
$CSCPUSH	Push Command Lines Below Current Pointer
$CSPOP	Pop The Command Stack
$CSPUSH	Push a Command Line Onto Command Stk
$CSPUSHN	Push Command Lines Onto Command Stack
$DISPCH	Display One Character
$DLMCHEK	Check Character For a Delimiter
$GETCHN	Get Response from CHAIN File / WS
$GETCHTO	Timeout for GETCSTK$ and GETCHN$
$GETCSTK	Get Response from Command Stack / WS
$GETCSTO	Timeout for GETCSTK$ and GETCHN$
$GETLINE	Get Response from Stack, CHAIN, or WS
$GETLNTO	Timeout Controlled Version of GETLINE
$KEYCHAR	Obtain One Translated Character
$KEYCLR	Clear the Keyin FIFO
$KEYIN	Accept a String From Keyboard to Memory
$KEYINTO	Timeout Controlled Version of KEYIN$
$LOGCLR	Clear Logging Flags
$LOGGING	Determine if Logging is Active
$LOGSET	Set Logging Flags
$PUTELGX	Log Message
$PUTELOG	Home-Down Roll, Log Error Message
$PUTERP	Home-Down Roll, Log/Display $MSG, Error
$PUTERPX	Log/Display from $MSG, Error
$PUTERRR	Home-Down Roll, Log/Display, Error
$PUTERRX	Log/Display Message, Error
$PUTLINE	Home-Down Roll, Display Message
$PUTLINX	Display Message
$PUTLNP	Home-Down Roll, Display from $MSG
$PUTLNPX	Display from $MSG
$PUTLOG	Home-Down Roll, Log Message
$PUTLOGX	Log Message
$PUTNOP	Home-Down Roll, Log/Display from $MSG
$PUTNOPX	Log/Display from $MSG
$PUTNOTE	Home-Down Roll, Log/Display Message
$PUTNOTX	Log/Display Message
$$RUN	Interface to $RUN System Call
$SETABTF	Set CHAIN Abort Flag

CROSS REFERENCED FUNCTIONS

Function names from Assembly Language RMS that have different names in DASL RMS.

BIGBCP$	Compare Large Strings; Assem Language UFR
BIGBT$	Move 16-bit Length; Assembly Language UFR
MFREMEM$	Deallocate a Block of Memory
MGETCLR$	Allocate a Cleared Block of Memory
MGETFST$	Allocate Free Space Block of Memory
MGETMEM$	Allocate a Block of Memory
MGETPAG$	Obtain a Page of Logical Memory Space
MMGINIT$	Initialize Memory Management
MRETPAG$	Release a Page of Logical Memory Space
WFUPDAT$	Update a Logical Record
WFUPDTL$	Update a Record in LOCATE Mode
WFWRITL$	Write a Record in LOCATE Mode

Types **TYPES and STRUCTURES**

*.........................means used by a FUNCTION as parameter type

DASL COMPILER DEFINED TYPES

BOOLEAN	scalar data type: 1 byte unsigned
BYTE	scalar data type: 1 byte unsigned
CHAR	scalar data type: 1 byte unsigned
INT	scalar data type: 2 bytes signed
LONG	scalar data type: 4 bytes signed
UNSIGNED	scalar data type: 2 bytes unsigned
FUNCTION	variable type: 13 parameters 1 result

D$FAR Include File

File Access Routines, REQUIRES D$RMS

	$ACB	Aim Control Block
	$DCB	Data File Control Block
*	*$FCBA*	File Control Block for AIM File
	$FCBAIM	Macro to Configure AIM File Control Blk
	$FCBAIMI	Initializer Macro Used by $FCBAIM
*	*$FCBD*	File Control Block for Direct File
	$FCBDIR	Macro to Configure Direct FCB
	$FCBDIRPRT	Macro used by Macros: $FCBPRT, $FCBDIR, $FCBDOVR
	$FCBDIRPRTI	Initializer Macro used by $FCBDIRPRT
	$FCBDOVR	Macro to Configure Direct-Overlapped I/O FCB
*	*$FCBIS*	File Control Block for ISAM File
	$FCBISAM	Macro to Configure ISAM FCB
	$FCBISAMI	Initializer Macro used by $FCBISAM
	$FCBPRT	Macro to Configure Print FCB
	$ICB	ISAM Control Block
	$MAB	Managed File Access Block
	$SCFMSCODES	FAR Exception Exit Codes

D$INC Include File

Basic DASL definitions, Standard Include

*	*D$CALLF*	Declare Name to be a Callable Function Type Routine	|*
	D$CCODE	Condition Code Flags	
	ILONG	24 Bit Number Structure, Integer	
*	*ULONG*	24 Bit Number Structure, Unsigned	|*

Types

*.........................	means used by a FUNCTION as parameter type

D$PCR Include File

Program Communications Region Definitions

	PCR	Program Control Region (externally defined)	

D$RMS Include File

Common Nucleus and UFR definitions

*	*$ENVN*	Environment Name	
	$ERRCODE	RMS Standard Error Code	|*
	$EXTT	File Extension	
*	*$FILEPTR*	File Pointer Structure	
*	*$FILESPK*	$SCANFLS File Specification	
	$HSI	Hierarchical Structure Information Array	
*	*$LNAMET*	Library Member Name	
	$LSN	Logical Sector Number	
	$NAMEEXT	Name, and Extension	
*	*$NAMEEXTENV*	Name, Extension, and Environment	
*	*$NAMET*	File Name Type	
*	*$OPENPT*	Open Parameter Table	
	$OPTION	$SCANFLS Option Specification	|*
	$OPTTAIL	$SCANOS Option List Terminator	
*	*$PACKPW*	Packed Password	
*	*$PFDB*	Physical File Descriptor Block	
	$PFDBBUF	PFDB Buffer List Entry	
*	*$SFENT*	Symbolic File Table	
	$SFNT	Symbolic File Name	
	$SONT	Symbolic Option Field Name	
*	*$STARTADR*	Starting Address Type	
*	*$TIME*	Time in Seconds Since Beginning of 1901	

D$RMSGEN Include File

RMS General System Function Definitions

	$DSTINFO	Daylight Savings Time Start/Stop Table	
*	*$INFOITEM*	Info Returned by $INFO	
	$IRUFLAGS	Resource Utilization Flags	
	$NODEFLAGS	Node Flags in $INFO	
	$RSRCFLAGS	$INFO Resource Status Flags	|*
*	*$SETTIMEP*	$SETIME Parameter Table	
*	*$SYSTIME*	System Time	
*	*$SYSTINFO*	System Time Information	|*

Types

*.........................means used by a FUNCTION as parameter type

D$RMSIO Include File

File Handling, Block I/O, Disk, Printer,Pipe

	$ACCODES	File Access Codes
*	***$FILEINFO***	Returned by $FILENAM Mode of $FILES
	$FILEKEY	File Key Structure
	$FILEKEYS	File Key List Array
*	***$FILESTBL***	Redefined $PFDB,Multi-File Sys Calls
*	***$OPENPTS***	Special $OPENENV ($OMCHECK, $OMREPAR Modes) |*
*	***$PIPEGENPT***	Pipe Generation Parameter Table
*	***$SECSTAT***	Block I/O Status Control
*	***$SECURETBL***	$SECURE Parameter Table
	$SFITABLE	$GETSFI Information

D$RMSPROG Include File

Program Loading and Execution Control

*	***$INTS***	Interupt State Table

D$RMSSTRUCT Include File

Disk Structure Definitions

$ABSHDR	Library Absolute Element Header Sector
$LIBENTRY	Library Directory Entry
$LIBSECTOR	Library Sector Formats
$LIBTYPE	Library Directory Entry Types
$PABENTRY	Relocatable Program ID Sector PAB Entry
$PABFLAGS	Relocatable Program ID Sector PAB Flags
$RELCODE	Relocatable Sector Type Codes
$RELEPN	Reclocatable Entry Point Member Entry
$RELEPNS	Relocatable Entry Point Member Sector
$RELLINE	Relocatable DEBUG Line Numbers Sector
$RELLNENT	Relocatable DEBUG Line Number Entry
$RELOBJ	Relocatable Object Code Sector
$RELPID	Relocatable Program ID Sector
$RELXDEF	Relocatable External Definition Sector
$RELXDENT	Relocatable External Definition Entry
$RELXFER	Relocatable Starting Address Sector
$RELXREF	Relocatable External Reference Entry
$RELXRENT	Relocatable External Reference Entry
$UABSECTOR	User Abend Header Sector Format

Types

*........................means used by a FUNCTION as parameter type

D$RMSWS Include File

		Workstation
*	***$WSCONF***	$WCONFIG Status Bits
	$WSCONF2	$WCONFIG 3rd Status Byte Flags
	$WSCONFDS	$WCONFIG 4 Byte Status Structure
*	***$WSIOMODE***	$WSIO Mode Bits
*	***$WSTAT***	$WSTATUS Status Bits

D$UFRENV Include File

		Environment Handling
		Requires D$RMSIO include if $$FILENAM function used
*	***$ENVT***	User Environment Table Entry
*	***$UNPACKPW***	Unpacked Password

D$UFRGEN Include File

		General Utility
*	***$CVSTBL***	$CVSTIME Output Area
*	***$DISTBL***	$DISORT Parameter Table
	FFMTABL$	File Format Table

D$UFRLIB Include File

		Library Manipulation
*	***$MEMBER***	Library Member Structure

D$UFRNQDQ Include File

		NQ/DQ UFR Definitions
	$NQDQITEM	NQDQ List Item
*	***$NQDQMSG***	NQDQ Message
*	***$NQDQSTAT***	NQDQ Statistics

D$UFRRLD Include File

		Relocating Loader
	$RLDEF	Relocatable Loader Definition Structure
	$RLFLAGS	Relocatable Loader Flags
*	***$RLNAME***	Relocatable Loader Name Type
*	***$RLPARAM***	Relocatable Loader Parameters
	$RLREF	Relocatable Loader Reference Work Area
	$RLUDRTNF	Relocatable Loader User Routine Types
	$RLUMEMAF	Relocatable Loader User Routine Types

Types
*.......................means used by a FUNCTION as parameter type

D$UFRSCAN Include File

Command Interpreter (Scanning)

	$CFGEND	$SCANCFG Keyword List Terminator
*	***$CFGHDR***	$SCANCFG Configuration Header
	$CFGKEY	$SCANCFG Keyword Parameters

D$UFRWFIO Include File

Work File I/O

*	***$WFCB***	Work File I/O Control Block
	$WFCBFLAG	Work File I/O Flags
	$WFCBFLAG2	Second Control Flag Byte Type
	$WFIOPRTN	Physical I/O Function Routine Type
	$WFIOTABPRTN	Pointers to Phys I/O Routines

DASL DEFINED FLAGS AND VALUES

It has been suggested that having the values (essentially ADJECTIVES, which give meaning to the variables, NOUNS) listed in one place may serve several purposes. Knowing the values that the RMS system variables use may give a perspective of scope to the overall system range and extent.

All words listed here will be listed in the WORDS section which should always be refered to for more definition. The WORDS section should also be considered most current, in that this section is a compilation from that and other sources.

Many of these groups will be listed in the description of the TYPES and FUNCTIONS using them.

MISCELLANEOUS

```
---------------------------------------------------------------
INCLUDE FILE..........: D$INC                                |*
         MAXINT       077777         DEFINE'd                |*
         MAXUNSIGNED  0177777        DEFINE'd                |*
         MAXLONG      017777777777   DEFINE'd                |*
         NIL          0              DEFINE'd                |*
         FALSE        000   Boolean value: 'false'           |*
         TRUE         001   Boolean value: 'true'            |*

INCLUDE FILE..........: D$PCR                                |*
        $NO           0116     'N'                           |*
        $YES          0131     'Y'                           |*
                                                             |*
INCLUDE FILE..........: D$RMS                                |*
        $NOADR        0177777  Indicate "NO ADDRESS GIVEN"   |*
                      USED IN: $FILESPK                      |*
        $NOPSK        0377  DEFINE'd Indicate "No PSK given" |*
                      USED IN: $PFDBUF.PSK                   |*
        $ENVTERM      0377  Environment data string terminator |*
                      USED IN: $OPENPT.$OTENV                |*
                               $PIPEGENPT.$PIPETERM          |*
        $MAXNPW       20    Max # of passwords in an env     |*
        $MSGLGT       81    Length of $MSG buffer            |*

INCLUDE FILE..........: D$RMSIO                              |*
        $FDTKEYN      9     Number of keys                   |*
                      USED IN: $FILEKEYS                     |*
        $FINDNOD      077   '?'find the node with the given  |*
                            resource                         |*
        $FOREVER      0377                                   |*
                      USED IN: $PFDB.$PTIMER                 |*
                               (Pipe and Rim)                |*
```

$AC... access codes $ACCODES

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: $OPENENV
USED IN TYPE [.FIELDS]: $ACB.$ACBA, $ACCODES, $DCB.$DCBACC,
                        $FILEKEY.$FDTACCO, $ICB.$ICBACC,
                        $OPENPT.$OTCODE, $PIPEGENPT.$PIPEIAC,
                        $SECURETBL.$SSFIAC

        $ACCODES  SET($NEWFILE,$ACREAD,$ACWRIT,$ACATALG,
                      $ACREATE,$ACRENM,$ACKILL,$ACSECQ);
        $NEWFILE  0001   New file
        $ACREAD   0002   Read data
        $ACWRIT   0004   Write data and deallocate space
        $ACATALG  0010   Obtain catalog information
        $ACREATE  0020   Create files under this catalog
        $ACRENM   0040   Rename the file
        $ACKILL   0100   Delete the file
        $ACSECQ   0200   Change security info
        $ACREPX   0200   Exclusive access,disk $OMREPAR only
        $ACMAX    0376   Sum of the $ACCODES
        $ACMAX    DEFINE($ACMAX,($ACSECQ+$ACKILL+$ACRENM
                         +$ACREATE+$ACATALG+$ACWRIT+$ACREAD))
```

A$..... AIM flags

```
---------------------------------------------------------------
INCLUDE FILE..........: D$FAR
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $FCBA.$FCBFLG2


        A$ABORT  0001 AIMDEX Aborted (Invalid Index)
        A$PMISS  0020 Primary record key must mis-match
        A$PRISEL 0040 Primary select active
        A$UPCASE 0100 Force upper case
        A$SHARE  0200 Shared AIM index
```

$BFT... used for / by Buffer Allocation Routines (FAR)

```
---------------------------------------------------------------
INCLUDE FILE..........: D$FAR
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: none

        $BFTINDX  0001  ISAM index buffer type
        $BFTDATA  0002  Data file buffer type
        $BFTOTHR  0003  Other buffer type
```

$CFG... $SCANCFG control flags

```
-------------------------------------------------------------
INCLUDE FILE..........: D$UFRSCAN
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $CFGKEY.$CFGFLAG

        $CFGNOVM  0017 (4 bits) Number of values moved
        $CFGLONG  0020 Value too large
        $CFGGTN   0040 Too many values
        $CFGVPT   0100 Value type error
        $CFGFND   0200 Keyword found
```

$CIL... logoff message numbers

```
-------------------------------------------------------------
INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: none

        $CILOG0   0000 Error in SIGNON
        $CILOG1   0001 Can not be loaded
        $CILOG2   0002 Command Interpreter can not be loaded
        $CILOG3   0003 Error in Command Interpreter
        $CILOG4   0004 Not enough memory for SIGNON
        $CILOG5   0005 SIGNON aborted
        $CILOG6   0006 LOGOFF key sequence
        $CILOG7   0007 LOGOFF forced from program
        $CILOG8   0010 ** UNUSED **
        $CILOG9   0011 Error in LOGOFF program
```

$CIS... CI status codes

```
-------------------------------------------------------------
INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: $RUN, $SINDEP
USED IN TYPE [.FIELDS]: none

        $CISNORM  0000  'NORMAL' system call exit
        $CISERRR  0001  'ERROR' system call exit
        $CISABT   0002  'ABORT' key sequence exit
        $CISUMV   0003  User mode violation exit
        $CISWSER  0004  Workstation error exit
        $CISVABT  0005  VANTAGE 'Abort' Key Sequence Exit
        $CISFMT   0006  Format error in progrm to execute
        $CISREAD  0007  Read I/O err in prgrm to execute
        $CISMEM   0010  Not enough mem for prgm execute
        $CISADR   0011  Insufficient logical address space
        $CISACC   0012  Memory access violation during LOAD
        $CISDUAL  0013  Dual Sector Tables not supported
        $CISHAR   0014  Shared program mis-match
        $CISYSTB  0015  Insufficient System Table space
```

$CK.... controller kinds

```
INCLUDE FILE..........: D$RMSGEN
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $INFOITEM.$ICKIND
```

$CK9350	0000	9350 Cartridge disk
$CK9370	0001	9370 25 MB Mass stg disk
$CK9374	0002	9374 10 MB Mass stg disk
$CK9390	0003	9390 67 MB Mass stg disk
$CK88D1	0004	8800 SMD/MMD disk IMOD
$CK88MEM	0005	8800 memory bank
$CK9301	0006	9301 20 MB Whizzie
$CK9310	0007	9310 10 MB Cynthia
$CK1403	0010	1403 Tortilla flex disk
$CK9315	0011	9315 10 MB Cyclone
$CK9324	0012	Moses Controller

$CM..... close modes

```
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: $ACLOSE, $CLOSE, $DCLOSE, $ICLOSE
USED IN TYPE [.FIELDS]: none
```

$CMUNCH	000	No change in file size
$CMSIZE	001	Deallocate to specified LSN
$CMCHOP	002	Deallocate to EOF LSN

$DEL.... Delimiters

```
INCLUDE FILE..........: D$UFRSCAN
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: Command lines, etc.
```

$DELMSYM	075	'='	Symbolic field name precedes and
$DELMEXT	057	'/'	File extension follows
$DELMENV	072	':'	Environment name follows
$DELMHSI	056	'.'	HSI level delimiter
$DELMQRY	077	'?'	Query mark
$DELMORE	053	'+'	More mark
$DELMNOT	055	'-'	Not mark
$DELMOPQ	042	'"'	Option quoted value delimiter
$DELMOPT	073	';'	Command line options delimiter
$DELMST1	054	','	Valid specification terminator

$DK..... resource kinds

INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $INFOITEM.$IROKIND $OPENPT.$OTKIND

$DKWS	000	Work station (pseudo res)
$DKDISK	001	Disk
$DKPIPE	002	Pipe (soft resource)
$DKPRINT	003	Printer
$DKCASS	004	Cassette tape
$DKMAGT	005	Industry compatible mag tape
$DKCOMM	006	Communications channel
$DKTIMER	007	Delay timer clk,soft res
$DKCARDR	010	Card reader
$DKCARDP	011	Card punch
$DKPTR	012	Paper tape reader
$DKPTP	013	Paper tape punch
$DK883M	014	8800 3M Cartridge tape
$DK863M	015	8600 3M Cartridge tape
$DKSMPLR	016	Task execution time sampler
$DKRIM	017	Direct RIM access
$DKFAX	020	FAX Equipment
$DKMAX	020	Largest resource kind number

$EOF.... $SECEOF modes

INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: $SECEOF
USED IN TYPE [.FIELDS]: none

$EOFGET	000	Get the current EOF LSN
$EOFSET	001	Set the current EOF LSN
$EOFWRIT	002	Write to EOF immediately

$FCS...

INCLUDE FILE..........:	D$FAR
USED IN FUNCTIONS.....:	none
USED IN TYPE [.FIELDS]:	$FCBA.$FCBFLG1, $FCBD.$FCBFLG1, $FCBIS.$FCBFLG1, $FCBIS.$FCBFLG2

$FCSIDUP	0001	Indicates duplicate keys are allowed
$FCSOVER	0001	Overlapped I/O
$FCSCMPR	0002	Compressed records
$FCSBIN	0004	Opened file is binary
$FCSOPEN	0010	This file is open
$FCSMNGD	0020	This file is managed at an FMT
$FCSTICB	0040	Primary ISAM FCB
$FCSTDCB	0100	DFCB (direct or byte)
$FCSTSIB	0140	Secondary ISAM FCB
$FCSTPRT	0200	Printer FCB - DISK res
$FCSTPRU	0240	Printer FCB - PRINT res
$FCSTACB	0300	Primary AIM FCB
$FCSTSAB	0340	Secondary AIM FCB
$FCSTMSK	0340	FCB type mask

$FFMT... file formats

```
------------------------------------------------------------------
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: $FILEFMT
USED IN TYPE [.FIELDS]: FFMTABL$.$FFTCODE, $FILEINFO.$FILEMT,
                        $OPENPT.$OTFMT, $SECURETBL.$SSFFMT
```

$FFMTSYS	0000	System File
$FFMTTMP	0001	Temporary File (Must be a 1)
$FFMTTXT	0002	Text (Logical Records)
$FFMTISM	0003	Isam Index
$FFMTL55	0004	Loadable 5500 REL/ABS Object Library
$FFMTRAC	0005	Non-Loadable REL/ABS Object Library
$FFMTDBC	0006	Databus Object Code
$FFMTBAS	0007	Basic Object Code
$FFMTMAC	0010	Macro Library
$FFMTWPS	0011	Word Processing Library
$FFMTJOB	0012	CHAIN Job File
$FFMTBIN	0013	Binary Data
$FFMTUTX	0014	Uncompressed Text Data
$FFMTMFD	0015	Managed File Descriptor
$FFMTUPF	0016	Universal Print Format
$FFMTWPF	0017	Word Processing Format
$FFMTAIM	0020	AIM format
$FFMTXFD	0021	Extended file (multi-volume file) |*
$FFMTL66	0024	Loadable 6600/8x00 product
$FFMTL80	0025	Loadable 8600/8800 product
$FFMTRPM	0040	Min val, released product types
$FFMTR55	0040	Released 5500/3800 product
$FFMTR66	0041	Released 6600/8x00 product
$FFMTR80	0042	Released 8600/8800 product
$FFMTPTR	0050	File-Pointer File

$FIL... File specification qualifying options

```
------------------------------------------------------------------
INCLUDE FILE..........: D$RMS                                   |*
USED IN FUNCTIONS.....: none                                    |*
USED IN TYPE [.FIELDS]: $FILESPK.$FSOOPT                        |*
                                                                |*
```

$FILNAMR	0001	Name required	|*
$FILEXTR	0002	Extension r quired	|*
$FILENVR	0004	Environment required	|*
$FILANYR	0010	Some field entry is required	|*
$FILFDEF	0020	The field is defined	|*
$FILQMOK	0040	Query, More, or Not marks allowed	|*
$FILNDSP	0100	Inhibit display if undefined	|*

$FILE... $FILES modes

INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: $FILES
USED IN TYPE [.FIELDS]: none

$FILEPCN	00	$FILES mode: Get file FDT-PCN's
$FILENAM	01	$FILES mode:Get file names from FDT-PCN's
$FILEGET	02	$FILES mode: Get file name from FAV
$FILECHK	03	$FILES mode: Check for file opened
$FILEDAT	04	Get FDT data for specific FAV

$g... Workstation Graphics Characters

INCLUDE FILE..........: D$WORKSTN |*
USED IN FUNCTIONS.....: $WSIO |*
USED IN TYPE [.FIELDS]: none |*

$gCurs	0000	Cursor: █	|*
$gVCurs	0001	Vantage cursor: ▯	|*
$gTLCorn	0002	Top left corner: ┌	|*
$gTRCorn	0003	Top right corner: ┐	|*
$gBLCorn	0004	Bottom left corner: └	|*
$gBRCorn	0005	Bottom right corner: ┘	|*
$gVBar	0006	Vertical bar: │	|*
$gHBar	0007	Horizontal bar: ─	|*
$gTTBar	0010	Top T-bar: ┬	|*
$gBTBar	0011	Bottom T-bar: ┴	|*
$gLTBar	0012	Left T-bar: ├	|*
$gRTBar	0013	Right T-bar: ┤	|*
$gCross	0014	Cross: ┼	|*
$gCursU	0015	Cursor Up: ↑	|*
$gCursD	0016	Cursor Down: ↓	|*
$gCursL	0017	Cursor Left: ←	|*
$gCursR	0020	Cursor Right: →	|*
$gEnter	0021	Enter: ↵	|*
$gCommand	0022	Command: □	|*
$gVTLCorn	0023	Vantage top left corner: ◸	|*
$gVTRCorn	0024	Vantage top right corner: ◹	|*
$gVBLCorn	0025	Vantage bottom left corner: L	|*
$gVBRCorn	0026	Vantage bottom right corner: ◿	|*
$gVTBar	0027	Vantage top bar: z	|*
$gVBBar	0030	Vantage bottom bar: _	|*
$gVLBar	0031	Vantage left bar: │	|*
$gVRBar	0032	Vantage right bar: ▐	|*
$gUserMeta	0177	User data meta character: ■	|*

$I... $INFO modes

INCLUDE FILE..........: D$RMSGEN
USED IN FUNCTIONS.....: $INFO
USED IN TYPE [.FIELDS]: none

$IRMFND	004	Find named multi file resource
$IRMFAL	005	Return all multi file resources
$IRSFND	006	Find named single file resource
$IRSFAL	007	Return all single file resources
$ITASKND	010	Find named task
$ITASKAL	011	Return all tasks
$INODEND	012	Find named available node
$INODEAL	013	Return all available nodes
$ILINKND	014	Find named connection link
$ILINKAL	015	Return all connection links
$ITASKME	016	Return caller's task info
$IMYNODE	017	Return name of local node
$IDLMTAB	020	Return delimiter table
$ISTARTT	021	Return system startup time
$ICONTV	022	Return all controller variables
$IRUDATA	023	Return resource utilization data
$ISPVND	024	Return named shared program variable
$ISPVAL	025	Return all shared program variable

$INF... Node Flags $NODEFLAGS

INCLUDE FILE..........: D$RMSGEN
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $INFOITEM.$ILFLAGS, $INFOITEM.$INFLAGS

$INFOFF	001	Offline: ANV disconnect; CLV hard error
$INFTXER	002	Transmitter blocked by error
$INFCONG	004	ANV in process of connection
$INFCFU	010	Checking file in use
$INFIFS	020	Incoming file support configured
$INFFMA	040	FAV Markers Available

$IRF... Resource status flags: $RSRCFLAGS

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSGEN                               |*
USED IN FUNCTIONS.....: none                                   |*
USED IN TYPE [.FIELDS]: $INFOITEM.$IROFLAG                     |*
                                                               |*
        $IRFOFF    0001 Resource off-line                      |*
        $IRFOCP    0002 Resource occupied                      |*
        $IRFWRP    0004 Resource write protected               |*
        $IRFCHK    0010 SYSCHECK in progress                   |*
        $IRFSTP    0020 Disk System Table problems             |*
        $IRFSPC    0040 Special Open Mode - Open if off-line   |*
        $IRF6      0100 ** UNDEFINED **                        |*
        $IRFBSD    0200  Resource is byte string device        |*
```

$IRU... Resource utilization flags: $IRUFLAGS

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSGEN                               |*
USED IN FUNCTIONS.....: none                                   |*
USED IN TYPE [.FIELDS]: $INFOITEM.$IRUFLAG                     |*

        $IRUx     0001   **UNUSED**                            |*
        $IRUIFH   0002   Incoming filehandler in use           |*
        $IRUOFH   0004   Outgoing filehandler in use           |*
        $IRUIFA   0010   Incomming file access supported       |*
```

$ISP... Shared program status

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSGEN                               |*
USED IN FUNCTIONS.....: none                                   |*
USED IN TYPE [.FIELDS]: $INFOITEM.$ISPSTAT                     |*
                                                               |*
        $ISPLOCK  0200 Shared program locked into memory       |*
        $ISPMEM   0037 Shared program PSK count                |*
```

$L... Logical codes in file

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSIO                                |*
USED IN FUNCTIONS.....: none                                   |*
USED IN TYPE [.FIELDS]: none                                   |*
                                                               |*
        $LMCV        0371  Minimum control character value     |*
        $LSPC        0371  Space compression count follows     |*
        $LEOR        0372  End of record mark                  |*
        $LEOF        0373  End of file mark                    |*
        $LST         0374  Special text mark                   |*
        $LEOB        0375  End of block mark                   |*
        $LXX         0376                                      |*
        $LDEL        0377  Deleted data mark                   |*
```

$LF... $LOCKFAV modes

```
INCLUDE FILE..........: D$RMSSPEC
USED IN FUNCTIONS.....: $LOCKFAV
USED IN TYPE [.FIELDS]: none
```

$LFLOKSP	000	Lock specified FAV
$LFULOKS	001	Unlock specified FAV

$LIB....   library member types

```
INCLUDE FILE..........: D$RMSSTRUCT
USED IN FUNCTIONS.....: $LBGTLSN
USED IN TYPE [.FIELDS]: $LIBENTRY.$LIBMTYP, $MEMBER.$LIBMTYP
```

$LIBFREE	000	Free entry
$LIBTERM	001	End of library
$LIBLINK	002	Link to nxt drctry sctor
$LIBABSX	003	"ABS" format executable
$LIBABSO	004	"ABS" format overlay
$LIBRELL	005	"REL" format
$LIBT006		
$LIBEPN	007	Entry point names
$LIBT008		
$LIBT009		
$LIBT010		
$LIBT011		
$LIBDLL	014	ARC down-line load format

```
USED IN TYPE [.FIELDS]: $ABSHDR.$LIBMAP
```

$LIBMT	000	Memory sector empty (not used)
$LIBPRIV	001	Memory sector private
$LIBSHAR	002	Memory sector shared
$LIBTBAD	003	Illegal mem sector type

$LIBMXPG	57	$LIBPG1 Array Size, maximum nbr of page groups |*

$MC..... memory control function

```
INCLUDE FILE..........: D$RMSMEM                                  |*
USED IN FUNCTIONS.....: $MEMCTL
USED IN TYPE [.FIELDS]: none

        $MCMDON    0  Activate memory diagnostic task
        $MCMDOFF   1  De-activate memory diagnostic task
        $MCMDTST   2  Perform memory diagnostic
        $MCDSON    3  Switch user task to dual sector mode
        $MCDSOFF   4  Switch user task to single sector mode
        $MCDSTST   5  Test sector table mode
```

$MPROT... $MEMPROT modes

```
INCLUDE FILE..........: D$RMSMEM
USED IN FUNCTIONS.....: $MEMPROT
USED IN TYPE [.FIELDS]: none

        $MPROTRW  0000  Set memory to read/write
        $MPROTRO  0200  Set memory to read only
```

$OM..... open modes

```
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: $AOPEN, $DOPEN, $IOPEN, $OPEN, $OPENENV
USED IN TYPE [.FIELDS]: $OPENPTS

        $OMREAD    000 Open mode: Shared read-only access
        $OMSHARE   001 Open mode: Shared read/write access
        $OMEXCL    002 Open mode: Exclusive read/write access
        $OMPREP    003 Open mode: Open or create file
        $OMCREAT   004 Open mode: Create a new file
        $OMCHECK   005 Open mode: Disk structure check access
        $OMREPAR   006 Open mode: Disk structure repair access
        $OMBYPAS   007 Open mode: Bypass passwrd/security chks
```

$OPTTERM $SCANOS terminator

```
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $OPTTAIL.$$OPTTERM

        $OPTTERM    0377  $SCANOS OPT Terminator
```

$OPTF... option flags

```
-------------------------------------------------------------------
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $OPTION.$OPTFLG

        $OPTFDEF    001  Returned if option given
        $OPTFVAL    002  Returned if value given
        $OPTFQOK    004  Option value may be quoted
```

$OPTV... option value flags

```
-------------------------------------------------------------------
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $OPTION.$OPTVAL

        $OPTVSET   0376  Option had no value
        $OPTVCLR   0377  Option not given
```

$PABF... PAB flags

```
-------------------------------------------------------------------
INCLUDE FILE..........: D$RMSSTRUCT                                 |*
USED IN FUNCTIONS.....: none                                        |*
USED IN TYPE [.FIELDS]: $PABENTRY.$PABTBFLG                         |*
                                                                    |*
        $PABF000  0001 Unassigned                                   |*
        $PABF001  0002 Unassigned                                   |*
        $PABFDATA 0004 Data PAB                                     |*
        $PABFCOMN 0010 Common PAB                                   |*
        $PABFPS   0020 PAB must not cross page boundry              |*
        $PABFTP   0040 PAB must start on page boundry               |*
        $PABFREL  0100 PAB is Relocatable                           |*
        $PABASS   0200 PAB Assigned                                 |*
```

$PCRAFGA abort flag

```
-------------------------------------------------------------------
INCLUDE FILE..........: D$RMS
USED IN FUNCTIONS.....: $SETABTF
USED IN TYPE [.FIELDS]: (PCR)$PCRABTF                               |*

        $PCRAFGA  000001 General abort
                                                                    |*
```

$PCRDF... command interpreter flags $PCRCMDF

```
------------------------------------------------------------
INCLUDE FILE..........: D$PCR
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: ($PCR)$PCRCMDF                     |*

        $PCRDFNH  000001  Inhibit signon/heading display
        $PCRDFEH  000002  Display all heading records
        $PCRDFSO  000004  Cmd int entered from signon program
        $PCRDFNW  000010  No workstation available
        $PCRDFFK  000020  Abort GETLINE$,KEYIN$ on function key
        $PCRDFCF  000040  Reset char font and translate table
        $PCRDFNC  000100  No STOP bar - clears HELP window    |*
        $PCRDFBJ  000400  Batched job facility is active
        $PCRDFWW  001000  This is version II command int
        $PCRDFWW  001000  Standard window was active
        $PCRDFWI  002000  Standard window is active
        $PCRDFMC  004000  CMD Line was Menu-Generated
        $PCRDFNS  010000  No STOP bar - does not clear window |*
        $PCRDFCW  020000  Current Window data valid
        $PCRDFML  040000  Menu line exists;
                          $PCRCLEL points to New Line
```

$PCRLF.. logging flags

```
------------------------------------------------------------
INCLUDE FILE..........: D$PCR
USED IN FUNCTIONS.....: $LOGCLR, $LOGSET
USED IN TYPE [.FIELDS]: (PCR)$PCRLOGF                      |*

        $PCRLFAC  0001   Logging active
        $PCRLFSP  0002   Logging suspended
        $PCRLFEO  0004   Log only error messages
        $PCRLFNI  0010   Log note display inhibited
        $PCRLFFO  0020   Log file open
        $PCRLFHR  0040   HD/RU before each logged message
```

$PRI... user task priority levels

```
------------------------------------------------------------
INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: $SETPRI
USED IN TYPE [.FIELDS]: $INFOITEM.$ITOPRTY

        $PRINORM   004  "NORMAL" priority level ($NRPRIOR/2)  |*
        $PRIMAX    007  "LOWEST" priority level
```

$PTF... tape, cassette subfunctions

```
INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $PFDB.$PSUBF                             |*
```

$PTFREAD	000	Block read
$PTFRDRV	001	Block read reverse
$PTFFSPB	002	Forward space (skip)to next block
$PTFBSPB	003	Backspace to previous block
$PTFFSPF	004	Forward space to next tape-mark
$PTFBSPF	005	Backspace to previous TAPE-MARK
$PTFRWND	006	Rewind the tape, GO READY
$PTFULOD	007	Rewind and unload the tape
$PTFWRIT	000	Block write
$PTFWRTM	001	Write a tape-mark
$PTFERAS	002	Erase some (3.5 inches) tape, Write extended inter-record gap |*

$RFITRC $RFI... trap remainder flag

```
INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: $RFIAKS, $RFIDKS, $RFIFK, $RFIKKS
USED IN TYPE [.FIELDS]: none
```

$RFITRC	0200	$RFI trap remain clear bit

$SEC... Block I/O status bits: $SECSTAT |*

```
INCLUDE FILE..........: D$RMSIO                                  |*
USED IN FUNCTIONS.....: $SECCHK, $SECWAIT                        |*
USED IN TYPE [.FIELDS]: none                                     |*
                                                                 |*
```

$SECWP	0001	Resource physical write protect	|*
$SECACT	0002	Operation still in progress	|*
$SECERR	0004	Soft error	|*
$SECTMD	0010	Tape-mark detected	|*
$SECFLOK	0020	FAV locked across CLOSEALLs ($LOCKFAV)	|*
$SECSTOP	0040	I/O stopped by user	|*
$SECSS	0100	Stop I/O has been sent	|*
$SECBSD	0200	Resource is a byte string device	|*

$SK..... resource subkinds

INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $INFOITEM.$IROSUBK, $OPENPT.$OTSUBK

FAX	$SKFCI	000	FAX communications interface
	$SKAFX	001	Peripheral FAX device
COMM	$SKCM481	001	Multi function com adaptor: 9481 MFCA
	$SKCM400	002	Async com adaptors: 9400, 9401, 9402
	$SKCM401	003	Async com 9400 with Bell 103 modem
	$SKCM402	004	Async com 9400 with Bell 202 modem
	$SKCM462	005	Port on multi port: 9462 MPCA port
	$SKCM38I	006	Internal com adaptor in 1800/3800
	$SKCM86I	007	Internal com adaptor in 8600
	$SKCM88I	010	Internal com adaptor in 8800
	$SKCM86M	011	Comm Device on 8600 MPCA
DISK	$SKDKFLX	001	Flexible diskette on 1800 Microbus
	$SKDKCTG	002	9350: 2.5 MB Cartridge on 5500/6600
	$SKDKM10	003	9374: 10 MB Mass storage on 5500/6600
	$SKDKM25	004	9370: 25 MB Mass storage on 5500/6600
	$SKDK67	005	9390: 67MB on 55/66MIDS or 8800 IMOD
	$SKDK134	006	9390 Disk: 134 MB fixed on 8800 IMOD
	$SKDKM20	007	9301:20MB fxd disk on 8600PIO "Whizzie"
	$SKDKC10	010	9310: 10 MB Cynthia on 8600 Microbus
	$SKDKF01	011	1403: 1 MB Dbl-Side Dbl-Dens Diskette "TORTILLA",on 8600 uBus
	$SKDKW10	012	9315: 10 MB Cyclone on 8600 Microbus
	$SKDKS10	013	10 MB Moses on 8600 Microbus
	$SKDKS40	014	40 MB Moses on 8600 Microbus
	$SKDKW20	015	20 MB Cyclone on 8600 Microbus |*
TAPE	$SKMT78	001	7-track 800 BPI Tape
	$SKMT98	002	9-track 800 BPI Tape
	$SKMT916	003	9-track 1600 BPI Tape
	$SKMT75	004	7-track 556 BPI Tape
NONE	$SKUNDEF	000	Resource has no subkind
PRINTER	$SKPTLOC	001	Line printer
	$SKPTFRE	002	Freedom printer
	$SKPTFST	003	Freedom prntr with secondary tractor
	$SKPTSVO	004	Servo printer
	$SKPTMER	005	Mercury printer parallel I/O
	$SKPT601	006	Mercury serial printer attached to MPCA
	$SKPT611	007	Orion serial printer attached to MPCA
	$SKPT621	010	Freedom serial printer attached to MPCA
	$SKPT297	011	132 Col Serial printer attached to MP

continued...

```
Note:  The following values are also used in type: $WSCONF      |*
       masked with $WSKMASK                                     |*

WS      $SKWSNA    000  Workstation not available
        $SKWS56    001  5500/6600 processor console
        $SKWS38    002  1800/3800 processor console
        $SKWS36    003  3601/8200 ver 1.1  Multiport Terminal
        $SKWSRMS   004  RMS WS (8200 ver.2 multi-port terminal)
        $SKWS86    005  8600 processor console
        $SKWS822   006  8220 workstation
        $SKWS823   007  8230 workstation                        |*
        $SKWSALN   010  Alien device (non Datapoint)
```

$SQL... security levels

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSGEN
USED IN FUNCTIONS.....: $SETSQL
USED IN TYPE [.FIELDS]: $OPENPT.$OTSQL, $SECURETBL.$SSFSL

        $SQLCHEK   010  Security level required to check
        $SQLMAX    011  Highest possible security level
        $SQLREPR   011,$SQLMAX:Security levl requird to repair
```

$SS..... $SECURE modes

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: $SECURE
USED IN TYPE [.FIELDS]: none

        $SSGET    000  File security get from FDT
        $SSPUT    001  File security put into FDT
        $SSGETX   002  Get extra info from FDT
        $SSPUTX   003  Put extra info into FDT
```

$STOP... $STOPIO modes

```
---------------------------------------------------------------
INCLUDE FILE..........: D$RMSIO
USED IN FUNCTIONS.....: $STOPIO
USED IN TYPE [.FIELDS]: none

        $STOPONE   0000  Stop I/O given by PFDB
        $STOPALL   0001  Stop all I/O
```

$TC.... task control modes

```
INCLUDE FILE..........: D$RMSTASK
USED IN FUNCTIONS.....: $TASKCTL
USED IN TYPE [.FIELDS]: none

        $TCLOKS   0  Force LOGOFF
        $TCAKS    1  Force abort
        $TCDKS    2  Force DISPLAY key sequence trap
        $TCKKS    3  Force KEYBOARD key sequence trap
        $TCFK     4  Force FUNCTION key trap
        $TCVLKS   5  Force VANTAGE LOG-OFF Key Seq Trap
        $TCVAKS   6  Force VANTAGE ABORT Key Seq Trap
```

$TMADJ... Time adjustment direction |

```
INCLUDE FILE..........: D$RMSGEN                             |*
USED IN FUNCTIONS.....: none                                 |*
USED IN TYPE [.FIELDS]: $SYSTINFO.$TMADJDR                   |*
                                                             |*
        $TMADJDN  0, Adjust Clock Down                       |*
        $TMADJUP  2, Adjust Clock Up                         |*
```

$TMDST... Daylight Savings Time adjustment direction |

```
INCLUDE FILE..........: D$RMSGEN                             |*
USED IN FUNCTIONS.....: none                                 |*
USED IN TYPE [.FIELDS]: $DSTINFO.$TMDSTFG                    |*
                                                             |*
        $TMDSTDI  1, from end. Note: 0= from start           |*
```

$UAB... User abend file

```
----------------------------------------------------------------
INCLUDE FILE..........: D$RMSSTRUCT
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $UABSECTOR

        $UABHDR    000  header sector LSN
        $UABALOC   001  Mem allocation table LSN
        $UABDUMP   003  Memory dump LSN

USED IN TYPE [.FIELDS]: $UABSECTOR.UABSTAT

        $UABDUAL   001  Dual Sector Tables active
        $UABPWSX   002  Executing in PWS Sector
        $UABPWSA   004  PWS Active
        $UABSHAR   010  Shared program active
        $UABLCL    020  local task

USED IN TYPE [.FIELDS]: $UABSECTOR.UABSPSK

        $UABPRO    001  PSK is read only
        $UABPPRV   002  PSK is private to this task
        $UABPSHR   004  PSK is shared
        $UABPDAD   010  PSK owned by father task
        $UABPPCR   020  PSK is the PCR sector
        $UABPPWS   032  PSK is the PWS Sector


INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: $USRABN
USED IN TYPE [.FIELDS]: none

        $UABSET    0  Activate User ABEND for this task
        $UABSETO   1  Activate User ABEND for other task
        $UABCLR    2  De-activate User ABEND for this task
```

$UTE... user task error codes

```
----------------------------------------------------------------
INCLUDE FILE..........: D$RMSPROG
USED IN FUNCTIONS.....: $GLUTEN
USED IN TYPE [.FIELDS]: $UABSECTOR.$UABERR

        $UTEWRIT 0000  UsrTsk Err:Memory write protect violat'n
        $UTEACCS 0001  User Tsk Err:Mem access protect violat'n
        $UTEINST 0002  Usr Tsk Err:Illegal Ins,usr mode violatn
        $UTEUNDF 0003  Usr Tsk Err:Undefined Ins or system call
        $UTEHALT 0377  User Tsk Err:Halt Ins for breakpointng
```

$WSBL Workstation Bottom Line

```
-----------------------------------------------------------------
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: $CURSON, $DISPCH, $GETCHN, $GETCHTO,
                        $GETCSTK, $GETCSTO, $GETLINE, $GETLNTO,
                        $GETPASS, $KEYCHAR, $PUTELGX, $PUTELOG,
                        $PUTERP, $PUTERPX, $PUTERRR, $PUTERRX,
                        $PUTLINE, $PUTLINX, $PUTLNP, $PUTLNPX,
                        $PUTLOG, $PUTLOGX, $PUTNOP, $PUTNOPX,
                        $PUTNOTE, $PUTNOTX, $WCONFIG, $WSIO
USED IN TYPE [.FIELDS]: $PCRCWVx(ver)

        $WSBL     000013  Bottom line
```

$WSLC & $WSRC horizontal cursor pos.

```
-----------------------------------------------------------------
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: $CURSON, $DISPCH, $GETCHN, $GETCHTO,
                        $GETCSTK, $GETCSTO, $GETLINE, $GETLNTO,
                        $GETPASS, $KEYCHAR, $PUTELGX, $PUTELOG,
                        $PUTERP, $PUTERPX, $PUTERRR, $PUTERRX,
                        $PUTLINE, $PUTLINX, $PUTLNP, $PUTLNPX,
                        $PUTLOG, $PUTLOGX, $PUTNOP, $PUTNOPX,
                        $PUTNOTE, $PUTNOTX, $WCONFIG, $WSIO
USED IN TYPE [.FIELDS]: $PCRCWHx(hor)

        $WSLC      0000  Left column
        $WSRC      0117  Right column
```

$WS... configuration status bits $WSCONF

```
INCLUDE FILE..........: D$RMS & D$RMSWS
USED IN FUNCTIONS.....: $WCONFIG
USED IN TYPE [.FIELDS]: $WSCONFDS.$WSCONC
```

```
        $WS0             1 Workstation Kind bit 0                |*
        $WS1             2 Workstation Kind bit 1                |*
        $WS2             4 Workstation Kind bit 2                |*
        $WS3           010 Workstation Kind bit 3                |*
        $WSKMASK       017 Workstation kind mask                 |*
        $WSCIAKA       020 INT & ATT keys downstrokes avail.
        $WSCKDKA       040 KBD & DPY keys static bits &
                           upstrokes avail
        $WSCIAUA      0100 INTERRUPT & ATTENTION keys static
                           bits and upstrokes available
        $WSCCKA       0200 Click available
        $WSBFSHA      0400 Shiftd function keys available
        $WSBFKUA     01000 F1 thru F5 upstrokes avail
        $WSBFKDA     02000 F1 thru F5 downstrokes avail
        $WSBFKSA     04000 F1 thru F5 static bits avail
        $WSBLCFA    010000 Display font set loadable
        $WSBIVA     020000 Inverted video available
        $WSBCPA     040000 Cursor positioning avail
        $WSBSWA     010000 Sub windows available
```

$WS2... $WCONFIG Third Status Byte Flags $WSCONF2

```
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $WCONFDS.$WSCON2
```

```
        $WS2POW     1   Tube just powered on
        $WS2IPL     2   System just Re-Booted
        $WS2CFL     4   Cursor font loadable
        $WS22LVA  010   2-Level video available
        $WS2ULNA  020   Underline available
        $WS2BNKA  040   Blink available
        $WS2EFKO 0100   Extended function keyboard on-line       |*
```

$WS3... $WCONFIG Third Status Byte Flags

```
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: none
USED IN TYPE [.FIELDS]: $WCONFDS.$WSCON3
```

$WSCON3 BYTE Values

```
        $WS3FD0    0  Character Font Not Loadable

                   $WS3FD1  thru $WS3FD4 : Table heading       |*
                      Size   Size  Length   Descriptor
                      Horiz  Vert  Per Char Required?
        $WS3FD1  1,   5      7      5         no                |*
        $WS3FD2  2,   5      7      7         no                |*
        $WS3FD3  3,   8      12     12        yes               |*
        $WS3FD4  4,   9      12     12        yes               |*


        $WS3FDMK   017 'Font Data' Mask
        $WS3NFMK  0360 'Number of Fonts' Mask
        $WS3NFS    004 'Number of Fonts' Shift Value
```

$WSM... $WSIO Mode Bits $WSIOMODE

```
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: $GETCHN, $GETCHTO, $GETCSTK, $GETCSTO,
                        $GETLINE, $GETLNTO, $KEYIN, $KEYINTO,
                        $WSIO
USED IN TYPE [.FIELDS]: PUTWSMD$
```

$WSMNW	0001	Inhibit 'DISPLAY' key wait
$WSMES	0002	Echo secret (char)
$WSMNI	0004	No case inversion
$WSMNE	0010	No echo or cursor
$WSMKCON	0020	Keyin continuous
$WSMDIGO	0040	Digits only
$WSMPADN	0100	Pad numeric decimal part
$WSMNESC	0200	No escape b4 0-037 or 0177

$WS... Status Bits $WSTAT

```
INCLUDE FILE..........: D$RMSWS
USED IN FUNCTIONS.....: $WSTATUS & $WAITIOS
USED IN TYPE [.FIELDS]: none
```

$WSF1	0001	'F1' key down
$WSF2	0002	'F2' key down
$WSF3	0004	'F3' key down
$WSF4	0010	'F4' key down
$WSF5	0020	'F5' key down
$WSDSP	0040	'DISPLAY' key down
$WSKBD	0100	'KEYBOARD' key down
$WSONL	0200	'ONLINE' status (always true)
$WSRDY	0400	Key ready
$WSINT	01000	'INTERRUPT' key down
$WSATT	02000	'ATTENTION' key down

$WS... Workstation Keyboard Codes

```
INCLUDE FILE..........: D$RMS & D$RMSWS
USED IN FUNCTIONS.....: $WSGETCH & $WSIO
                        (and UFRs: $KEYIN & $KEYINTO)          |*
USED IN TYPE [.FIELDS]: none
```

CODES Returned by $WSGETCH and $WSIO on: |*
|*

● 'STANDARD' keyboards |*
|*

● 'UNIVERSAL' keyboards |*
|*

● 'EXTENDED FUNCTION' keyboards |*

$WSRPTK (Repeated Key) Returned only by $WSGETCH (not $WSIO) |*

STANDARD Keyboard Codes: |*

Marking:

$WSBSPK	0010	'BACKSPACE' key
$WSCANK	0030	'CANCEL' key
$WSENTK	0015	'ENTER' key
$WSECHO	0052	'*' used in keyin echo

Non-Marking:

$WSBLANK	0040	' ' , Reserved for blank code
$WSCURS	0000	Reserved for cursor code (1800)
$WSDELK	0177	'DEL' key (used by katakana)

continued...

```
SPECIAL Keyboard Codes:                                              |*

Non-keystroke codes:
        $WSBADPK  0220 Bad parity received
        $WSKFULL  0234 Keyin fifo full
        $WSDLLR   0235 'DLL response (8220)' character               |*
        $WSTIMEO  0376 Last char in string if timed-out
        $WSWAKEK  0377 Wake up

Function Key Downstroke Codes:
        $WSDSPK    0200  'DISPLAY' key
        $WSKBDK    0202  'KEYBOARD' key
        $WSF1K     0204  'F1' key
        $WSF2K     0206  'F2' key
        $WSF3K     0210  'F3' key
        $WSF4K     0212  'F4' key
        $WSF5K     0214  'F5' key
        $WSINTK    0216  'INTERRUPT' key
        $WSATTK    0217  'ATTENTION' key

Function Key Upstroke Codes:
        $WSDSPUP   0201  'DISPLAY' key released
        $WSKBDUP   0203  'KEYBOARD' key released
        $WSF1UP    0205  'F1' key released
        $WSF2UP    0207  'F2' key released
        $WSF3UP    0211  'F3' key released
        $WSF4UP    0213  'F4' key released
        $WSF5UP    0215  'F5' key released
        $WSINTUP   0221  'INTERRUPT' key released
        $WSATTUP   0222  'ATTENTION' key released

Shifted Function Key Downstroke Codes:
        $WSF1KS    0223  'F1' key shifted
        $WSF2KS    0224  'F2' key shifted
        $WSF3KS    0225  'F3' key shifted
        $WSF4KS    0226  'F4' key shifted
        $WSF5KS    0227  'F5' key shifted
        $WSINTKS   0230  'INTERRUPT' key shifted
        $WSATTKS   0231  'ATTENTION' key shifted
        $WSDSPKS   0232  'DISPLAY' key shifted
        $WSKBDKS   0233  'KEYBOARD' key shifted

Last Static Key Code:
(for value testing)                                                  |*
        $WSLSTSK   0222  $WSATTUP  Last static key code
```

continued...

```
CODES Returned by $WSGETCH and $WSIO on:                          |*
                                                                  |*
●  'EXTENDED FUNCTION' keyboard only                              |*
                                                                  |*
Non-keystroke codes:
        $WSGENUP 0340 generic rollover key upstroke               |*
        $WSEFKFO 0341 extended function keyin FIFO overflow       |*
        $WSEFBPK 0342 Bad parity keycode - E.F. keyboard          |*

Defined Function Key Downstroke Codes:                            |*
        $WSSYSD  0200 'SYSTEM' key down                           |*
        $WSVIEWD 0202 'VIEW' key down                             |*
        $WSQUITD 0204 'QUIT' key down                             |*
        $WSPNTD  0206 'POINT' key down                            |*
        $WSUNDOD 0210 'UNDO' key down                             |*
        $WSHELPD 0212 'HELP' key down                             |*
        $WSCOPYD 0214 'COPY' key down                             |*
        $WSREMVD 0216 'REMOVE' key down                           |*
        $WSINSTD 0220 'INSERT' key down                           |*
        $WSRECLD 0222 'RECALL' key down                           |*

Defined Function Key Upstroke Codes:                              |*
        $WSSYSU  0201 'SYSTEM' key up                             |*
        $WSVIEWU 0203 'VIEW' key up                               |*
        $WSQUITU 0205 'QUIT' key up                               |*
        $WSPNTU  0207 'POINT' key up                              |*
        $WSUNDOU 0211 'UNDO' key up                               |*
        $WSHELPU 0213 'HELP' key up                               |*
        $WSCOPYU 0215 'COPY' key up                               |*
        $WSREMVU 0217 'REMOVE' key up                             |*
        $WSINSTU 0221 'INSERT' key up                             |*
        $WSRECLU 0223 'RECALL' key up                             |*

Numbered Function Key Downstroke Codes:                           |*
        $WSEFF1D 0230 function key '1' down                       |*
        $WSEFF2D 0232 function key '2' down                       |*
        $WSEFF3D 0234 function key '3' down                       |*
        $WSEFF4D 0236 function key '4' down                       |*
        $WSEFF5D 0240 function key '5' down                       |*
        $WSEFF6D 0242 function key '6' down                       |*
        $WSEFF7D 0224 function key '7' down                       |*
        $WSEFF8D 0226 function key '8' down                       |*

Numbered Function Key Upstroke Codes:                             |*
        $WSEFF1U 0231 function key '1' up                         |*
        $WSEFF2U 0233 function key '2' up                         |*
        $WSEFF3U 0235 function key '3' up                         |*
        $WSEFF4U 0237 function key '4' up                         |*
        $WSEFF5U 0241 function key '5' up                         |*
        $WSEFF6U 0243 function key '6' up                         |*
        $WSEFF7U 0225 function key '7' up                         |*
        $WSEFF8U 0227 function key '8' up                         |*
```

continued...

```
Geometric Symbol Function Key Downstroke Codes:                |*
        $WSSQARD 0244 'SQUARE' key down                        |*
        $WSTRIAD 0246 'TRIANGLE' key down                      |*
        $WSCIRCD 0250 'CIRCLE' key down                        |*

Geometric Symbol Function Key Upstroke Codes:                  |*
        $WSSQARU 0245 'SQUARE' key up                          |*
        $WSTRIAU 0247 'TRIANGLE' key up                        |*
        $WSCIRCU 0251 'CIRCLE' key up                          |*

Cursor Pad Key Downstroke Codes:                               |*
        $WSCKULD 0252 Cursor Key (UP-LEFT) down                |*
        $WSCKUCD 0254 Cursor Key (UP-CENTER) down              |*
        $WSCKURD 0256 Cursor Key (UP-RIGHT) down               |*
        $WSCKCLD 0260 Cursor Key (CENTER-LEFT) down            |*
        $WSCKCCD 0262 Cursor Key (CENTER-CENTER) down          |*
        $WSCKCRD 0264 Cursor Key (CENTER-RIGHT) down           |*
        $WSCKDLD 0266 Cursor Key (DOWN-LEFT) down              |*
        $WSCKDCD 0270 Cursor Key (DOWN-CENTER) down            |*
        $WSCKDRD 0272 Cursor Key (DOWN-RIGHT) down             |*

Cursor Pad Key Upstroke Codes:                                 |*
        $WSCKULU 0253 Cursor Key (UP-LEFT) up                  |*
        $WSCKUCU 0255 Cursor Key (UP-CENTER) up                |*
        $WSCKURU 0257 Cursor Key (UP-RIGHT) up                 |*
        $WSCKCLU 0261 Cursor Key (CENTER-LEFT) up              |*
        $WSCKCCU 0263 Cursor Key (CENTER-CENTER) up            |*
        $WSCKCRU 0265 Cursor Key (CENTER-RIGHT) up             |*
        $WSCKDLU 0267 Cursor Key (DOWN-LEFT) up                |*
        $WSCKDCU 0271 Cursor Key (DOWN-CENTER) up              |*
        $WSCKDRU 0273 Cursor Key (DOWN-RIGHT) up               |*

Miscellaneous Key Downstroke Codes:                            |*
        $WSCMDD  0274 'COMMAND' key down                       |*
        $WSENT1D 0276 'ENTER (LEFT)' key down (U.S.A.)         |*
        $WSNPE1D 0300 number pad 'ENTER' key (top) down        |*
        $WSBAK1D 0302 'BACKSPACE (LEFT)' key down (U.S.A.)     |*
        $WSSHF2D 0304 'SHIFT (LEFT/RIGHT)' key down (U.S.A.)   |*
        $WSSHF4D 0306 'SHIFT (RIGHT/RIGHT)' key down (U.S.A)   |*
        $WSALT2D 0310 'ALT (LEFT/RIGHT)' key down              |*
        $WSALT3D 0312 'ALT (RIGHT/LEFT)' key down              |*
        $WSENT2D 0314 'ENTER (RIGHT)' key down (U.S.A.)        |*
        $WSENTD  0314 $WSENT2D  'ENTER' key down (U.S.A.)      |*
        $WSBAK2D 0316 'BACKSPACE (RIGHT)' key down (U.S.A.)    |*
        $WSBAKSD 0316 'BACKSPACE' key down (U.S.A.)            |*
        $WSTABD  0320 'TAB' key down                           |*
```

continued...

```
Miscellaneous Key Upstroke Codes:                                 |*
        $WSCMDU  0275 'COMMAND' key up                            |*
        $WSENT1U 0277 'ENTER (LEFT)' key up (U.S.A.)              |*
        $WSNPE1U 0301 number pad 'ENTER' key (top) up             |*
        $WSBAK1U 0303 'BACKSPACE (LEFT)' key up (U.S.A.)          |*
        $WSSHF2U 0305 'SHIFT (LEFT/RIGHT)' key up (U.S.A.)        |*
        $WSSHF4U 0307 'SHIFT (RIGHT/RIGHT)' key up (U.S.A.)       |*
        $WSALT2U 0311 'ALT (LEFT/RIGHT)' key up                   |*
        $WSALT3U 0313 'ALT (RIGHT/LEFT)' key up                   |*
        $WSENT2U 0315 'ENTER (RIGHT)' key up (U.S.A.)             |*
        $WSENTU  0315 $WSENT2U  'ENTER' key up (U.S.A.)           |*
        $WSBAK2U 0317 'BACKSPACE (RIGHT)' key up (U.S.A.)         |*
        $WSBAKSU 0317 'BACKSPACE' key up (U.S.A.)                 |*
        $WSTABU  0321 'TAB' key up                                |*

Number Pad Key Downstroke Codes:                                  |*
        $WSNPTBD 0322 number pad 'TAB' key down                   |*
        $WSNPE2D 0324 number pad 'ENTER' key (bottom) down        |*
        $WSNPEND 0324 number pad 'ENTER' key down                 |*

Number Pad Key Upstroke Codes:                                    |*
        $WSNPTBU 0323 number pad 'TAB' key up                     |*
        $WSNPE2U 0325 number pad 'ENTER' key (bottom) up          |*
        $WSNPENU 0325 number pad 'ENTER' key up                   |*

LOCK and CASE INVERT Key Codes:                                   |*
        $WSLOCKD 0326 'LOCK' key down                             |*
        $WSCASED 0326 'CASE INVERT' key down                      |*

        $WSLOCKU 0327 'LOCK' key up                               |*
        $WSCASEU 0327 'CASE INVERT' key up                        |*

Control Key Downstroke Codes:                                     |*
        $WSSHFLD 0330 'SHIFT (LEFT)' key down (U.S.A.)            |*
        $WSSHF1D 0330 'SHIFT (LEFT/LEFT)' key down (U.S.A.)       |*
        $WSSHFRD 0332 'SHIFT (RIGHT)' key down (U.S.A.)           |*
        $WSSHF3D 0332 'SHIFT (RIGHT/LEFT)' key down (U.S.A.       |*
        $WSALTLD 0334 'ALT (LEFT)' key down                       |*
        $WSALT1D 0334 'ALT (LEFT/LEFT)' key down                  |*
        $WSALTRD 0336 'ALT (RIGHT)' key down                      |*
        $WSALT4D 0336 'ALT (RIGHT/RIGHT)' key down                |*

Control Key Upstroke Codes:                                       |*
        $WSSHFLU 0331 'SHIFT (LEFT)' key up (U.S.A.)              |*
        $WSSHF1U 0331 'SHIFT (LEFT/LEFT)' key up (U.S.A.)         |*
        $WSSHFRU 0333 'SHIFT (RIGHT)' key up (U.S.A.)             |*
        $WSSHF3U 0333 'SHIFT (RIGHT/LEFT)' key up (U.S.A.)        |*
        $WSALTLU 0335 'ALT (LEFT)' key up                         |*
        $WSALT1U 0335 'ALT (LEFT/LEFT)' key up                    |*
        $WSALTRU 0337 'ALT (RIGHT)' key up                        |*
        $WSALT4U 0337 'ALT (RIGHT/RIGHT)' key up                  |*
```

continued...

```
First and Last SHIFT/Control keycode values:                |*
(for value testing)                                         |*
        $WSSHFTF 0330 first SHIFT key                       |*
        $WSSHFTL 0337 $WSALTRU, last SHIFT key              |*
        $WSEFCTT 0337 E.F. keyboard - top control value     |*
```

$WS... $WSIO control codes

```
------------------------------------------------------------
INCLUDE FILE..........: D$RMSWS & D$RMS                    |*
USED IN FUNCTIONS.....: $WSIO
USED IN TYPE [.FIELDS]: none

Note:   Some of the control codes require a one byte
        value following the code, indicated (). Some
        require an address, indicated (()).  Some require
        some of each.

$WSIO String Control Code Range:

        $WSIOFCF  0200  $WSIO function code first value
        $WSIOFCL  0276  $WSK1CHR; $WSIO func code last value  |*

CURSOR CONTROL
        $H        0234  New cursor column follows (pos)
        $V        0235  New cursor row follows (pos)
        $HA       0236  Cursor column adjustment follows (adj)
        $VA       0237  Cursor row adjustment follows (adj)
        $CP       0240  Cursor position follows (vert),(horz)
        $HU       0241  Home up to upper left-hand corner
        $HD       0242  Home down to lower left-hand corner
        $NL       0243  Advance to new line
        $WSCURON  0214  Turn Cursor On at current position
        $WSCUROF  0215  Turn Cursor Off at current position

GENERAL
        $EEOF     0200  Erase from cursor to end of frame
        $EEOL     0201  Erase from cursor to end of line
        $RU       0202  Roll screen up one line
        $RD       0203  Roll screen down one line
        $ES       0231  End of string
        $EL       0232  Advance to new line,terminate strg
        $ESNF     0271  End Of String, don't flush display
        $NS       0233  New string address follows ((loc))
        $WSNOP    0244  No operation

        $WSSMODE  0245  Set mode (bits)
        $WSCMODE  0246  Clear mode (bits)

        $WSCKF    0247  Clear keyboard fifo

        $WSBEEP   0204  Beep
        $WSCLICK  0205  Click

        $WSIKCON  0210  Key click on
        $WSIKCOF  0211  Key click off
```

continued...

```
        $WSATTEN 0216  Enable KDS 3 Attributes; underline &
                       2-level video on 8600 console.  Has
                       no effect on other workstations.
        $WSCONFD 0272  WS Config data (Len),((Loc))
        $WSRECON 0273  WS Reconfig data (Len),((Loc))
                       where ((Loc)) has the format:
                       Mask_0, Value_0, Mask_1, Value_1
                       Mask_[Len-1], Value_[Len-1]           |*

VIDEO
        $WSVI    0206  Video inverted
        $WSVN    0207  Video normal
        $WSSVMOD 0264  Set video mode (mode)

        VIDEO MODES follow $WSSVMOD:
            $WSVM2L  0000  Vid Mode: Bold-face,
                           double intensity
            $WSVMUNL 0001  Video Mode: Underline
            $WSVMBNK 0002  Video Mode: Blink
            $WSVMAF  0003  Video Mode: Alternate font

PRINTER

        $WSPTRON 0212  Turn On Printer connected to WS
        $WSPTOFF 0213  Turn Off Printer connected to WS

        $WSLF    0265  Line feed for WS serial printers
        $WSFF    0266  Form feed for WS serial printers
        $WSCR    0267  Carriage return WS serial printers

INSERT DELETE OPEN CLOSE SCROLL WINDOW

        $WSRESET 0217  Reset Window to Default Screen size
        $WSRESTR 0225  Same as $WSRESET except 8600 KDS
                       attributes are not disabled

        $WSSWTB  0222  Set sub window (vert-top),(vert-bot)
        $WSSWLR  0223  Set sub window (horz-left),(horz-right)

        $WSIDOCS 0224 (INS, DEL, OPEN, CLOSE, SCROLL codes)

        Codes following $WSIDOCS:
           $WSINSCH 0  Insert space under cursor, shift down
                       (vert),(horz)                         |*
           $WSDELCH 1  Delete char under cursor, shift up
                       (vert),(horz)                         |*
           $WSINSLN 2  Roll down lines from cursor to bottom
           $WSDELLN 3  Delete line under cursor and roll up
           $WSOPENL 4  Open line from under cursor rolling
           $WSCLOSL 5  Close line from under cursor rolling
```

continued...

The scroll commands are followed by the characters

$WSSCRL	0006	Scroll left < followed by data >	
$WSSCRR	0007	Scroll right < followed by data >	
$WSSCRE	0010	End of scroll data	

OUT and IN Strings

$WSOS	0250	Output string (len),((loc))	
$WSONCH	0251	Output repeated (char),(n) times	
$WSIS	0252	In string (con),(max),((loc)),(end)	
$WSISI	0253	In string imm (con),(max),(skip),(end)	
$WSIN	0254	In numeric (lmax),(rmax),((loc)),(end)	
$WSINI	0255	In numrc imm (lmax),(rmax),(skip),(end)	
$WSITIME	0256	Set inter-char timeout to (t) seconds	

KEYIN TRANSLATE TABLE:

$WSSKXTA	0257	Set keyin translate table at ((loc))	
$WSSKXTP	0261	Set keyin xlate table at ((loc)),(psk)	
$WSKEYCH	0270	Keyin un-xlated character at ((Loc))	
$WSK1CHR	0276	Keyin un-xlated char at ((Loc))	|*
		with cursor on/off	|*
$WSECHOS	0262	Set echo secret disply char (char)	
$WSTWAIT	0263	Perform n second wait (n)	

CHARACTER FONT SET:

$WSLCFS	0260	Load char font set from ((loc))	
$WSCURFL	0274	Load cursor font from ((Loc))	
$WSCURDF	0275	Return to default cursor font	

continued...

```
$WSIO ESCAPE SEQUENCE 1 codes                              |*
for the Extended Function Keyboard                         |*
=====================================================
$WSESC1  0226 indicates that $WSIO 'escape-sequence-1'     |*
              codes follow.                                |*

Last control code values (for testing):                    |*
        $WSESC1L 1, $WSEFCTL;Last escape-seq-1 control code. |*
        $WSEFCL  016, $WSEFST2;Last EFK Control code value.  |*

These codes may follow $WSESC1:                            |*
        $WSDSCNT 0 Disconnect datastation                  |*
        $WSEFCTL 1 Expanded Function Keyboard control.     |*
             See the Escape-Sequence-1 codes that          |*
             follow this code; below...                    |*
-----------------------------------------------------
Escape 1 Sequences that may be used via:                   |*
                                                           |*
        o  Keycode Translate Module path, or               |*
        o  General Purpose Keyboard Emulation path.        |*

All sequences begin: $WSESC1,$WSEFCTL,...                  |*
-----------------------------------------------------
                                                           |*
      ...$WSEFRST 0 Same as $WSRESET except reset all but  |*
                    Keycode Xlate Module path              |*
                                                           |*
      ...$WSEFLOW 1 followed by (CONTROL);                 |*
                    Expanded function KBD Flow control     |*
                                                           |*
   CONTROL bit definitions:                                |*
        $WSLENON  001 Enable Keycode Translate Module      |*
                      path; nicknamed 'LENS'               |*
        $WSPUPOF  002 Disable Locked Key Processing;       |*
                      nicknamed 'PUPIL'                    |*
        $WSDKPOF  004 Disable Dead Key Processor           |*
        $WSBIFDT  010 Get data from Internal Buffer        |*
                      Module; nicknamed 'BIFOCAL'          |*
        $WSLNTOF  020 Disable Keycode Translate Module     |*
                      path; nicknamed 'LENS'               |*
                                                           |*
     ...$WSEFKID   3  followed by ((LOC)); Get keyboard I.D. |*
     ...$WSEFSTC   7  followed by ((LOC)); Get current static |*
                      bits, i.e. control codes.            |*
                                                           |*
     ...$WSEFSTF 010  followed by ((LOC)); Get control key |*
                      static bits from static key FIFO.    |*
                      LOC is six bytes long.               |*
                                                           |*
     ...$WSEFSTL 011  followed by ((LOC)) Get latched static |*
                      bits, i.e. the cumulative result of any |*
                      keys pushed since last occurrence of |*
                      control sequence.                    |*
```

continued...

```
  ...$WSEFST2 016 followed by ((LOC)); Get status bits    |*
                  + $WSTATUS bits. LOC is nine bytes; the |*
                  same 7 bytes for $WSEFSTC plus the 2    |*
                  bytes normally returned by $WSTATUS.    |*
  -----------------------------------------------------
  Escape 1 Sequences that require the                     |*
                                                          |*
     o  Keycode Translate Module path be enabled.         |*
  -----------------------------------------------------
                                                          |*
  ...$WSEFLCD 2 followed by (MASK),(CONTROL)              |*
                LCD display segment control               |*

MASK and CONTROL bit definitions (are the same):          |*

        $WSLCDLC  0001 LOWER CASE "a" (0/1) (OFF/ON)      |*
        $WSLCDUC  0002 UPPER CASE "A" (0/1) (OFF/ON)      |*
        $WSLCDDP  0004 DISPLAY (0/1) (OFF/ON)             |*
        $WSLCDKD  0010 KEYBOARD (0/1) (OFF/ON)            |*
        $WSLCDSQ  0020 SQUARE (0/1) (OFF/ON)              |*
        $WSLCDTR  0040 TRIANGLE (0/1) (OFF/ON)            |*
        $WSLCDCR  0100 CIRCLE (0/1) (OFF/ON)              |*

  ...$WSEFRPT   4 followed by (TIME); set repeat key      |*
                  timeout in increments of 16             |*
                  milliseconds.                           |*

  ...$WSEFECI   5 Enable case inversion                   |*
  ...$WSEFDCI   6 Disable case inversion                  |*

  ...$WSEFKAB 012 followed by (CKEY); Abort field keyin   |*
                  on detection of ocntrol key downstroke. |*

  ...$WSEFKAR 013 Reset all field keyin abort keys.       |*

  ...$WSEFKTP 014 followed by (CKEY); Activate function   |*
                  trap for control key downstroke         |*
                  specified.                              |*

  ...$WSEFKTR 015 Reset all function key traps.           |*
```

$XCFMS..

FMS exception codes: $SCFMSCODES

INCLUDE FILE..........: D$FAR
USED IN FUNCTIONS.....: $ADELCR, $APOS, $AREAD, $AREADCR, $AREADKG, $ARWRTCR, $AWRITE, $DDEL, $DDELCR, $DGETCRK, $DPOS, $DPOSNX, $DPOSPV, $DREAD, $DREADCR, $DREADNX, $DREADPV, $DRWRT, $DRWRTCR, $DWRITE, $IDEL, $IDELCR, $IDELK, $IINS, $IPOS, $IPOSKP, $IPOSKS, $IREAD
USED IN TYPE [.FIELDS]: none

$XCFMS00	000	Dummy code, Undefined
$XCFMS01	001	File pos out of range
$XCFMS02	002	No such record
$XCFMS03	003	ISAM key not found
$XCFMS04	004	ISAM duplicate key
$XCFMS05	005	Record already exists
$XCFMS06	006	No current record exists
$XCFMS07	007	File positioned to EOF

FUNCTION FORMATS

Description

```
 ------------------------------------------------
|               Some DEFINITIONS                 |
|      DASL: Datapoint Adavanced System Language |
|       RMS: Resource Management System          |
|     MACRO: ...a combining form,  meaning long  |
|  FUNCTION: ...a variable quantity dependent    |
|              upon other quantities             |
 ------------------------------------------------
```

DEFINE FUNCTION FIELDS

Definition of Function Format Fields

```
------------------------------------------------
DATE ENTERED: When added to Dictionary
UPDATED.....: Last change to Function or Description

CATEGORY: Function Kind

FILE....: DASL INCLUDE File  where
          Function is Declared

SYNTAX..: DASL macro statment of FUNCTION
          (with list of Arguments)

RESULT..: Function's resultant value, if any

USE.....: Syntax to use if function use requires
          only one condition test (which is not
          an error message condition)

ASSIGN..: Syntax to use if function use requires
          assigning function result to a variable
          for multiple condition tests.

ENTRY.1.: Argument #1 NAME and TYPE (if it is an
ENTRY.2.:          #2                entry parameter)
ENTRY.9.:     upto #9
```

```
..EXIT 1: Argument #1 NAME and TYPE (if it is an
..EXIT 2:          #2                exit parameter)
..EXIT 9:     upto #9
```

IF ERROR: Syntax to use if the function has an RMS error code condition. Sometimes this is the complete statement form required for use unless an ASSIGN: statement or group of statments is specified, in which case the IF ERROR: statement is used in sequence with them.

```
      -------------------------------------
     | NOTE:                                |
     | In the FUNCTION Section description  |
     | of error codes, contents of:         |
     |   $ERRC.$FUNC = SC$.... or           |
     |                 $UEC...              |
     |   $ERRC.$CODE = $EC...nn  or         |
     |                 $UEC...nn            |
     | The $EC..nn or $UEC..nn ends with a  |
     | DECIMAL NUMBER.                      |
     |       Statements testing the         |
     | value of $ERRC.$CODE may use the     |
     | word if it is defined in DASL.       |
     | Defined values will have :D$fileName |
     | or :* (if defined in D$ERRCODE)      |
     | following the line of description.   |
     | If the value is not defined, use the |
     | decimal value, not the word.         |
      -------------------------------------
```

REMARKS.: Miscellaneous Discription

SEE ALSO: Cross Referencing and Association of Functions

DASL DOC: The DASL Document, March 1982 (page) Not to Be MAINTAINED in released version

SPRM REF: RMS System Programmer's Reference Manual Not to Be MAINTAINED in released version

DATE of SPRM Referenced is SEPT 82

Examples of Possible Field Entrys

```
------------------------------------------------------
CATEGORY: User Function Routine
          File Access Routine
          System Call
          DASL External Function
          DASL Compiler Macro
          DASL Include Macro
          DASL Control Word
          DASL Declaration Word
          DASL Reserved Word
          Cross Reference


                               Functions used from
                               these files cause
                               additional REL Code
                               added to program.
                               __________________
                              |                  |
FILE....: D$INC *   D$RMSMEM     D$UFRENV   D$FAR
          D$RMS *   D$RMSSTRUCT  D$UFRERR
        1 D$ERRCODE D$RMSIO      D$UFRNUM
        1 D$ERRNUM  D$RMSWS      D$UFRSYS
        2 D$PCR     D$RMSPROG    D$UFRSCAN
        2 D$WORKSTN D$RMSGEN     D$UFRWS             |*
                    D$RMSSPEC    D$UFRGEN
         |_______|  D$RMSTASK    D$UFRLIB
        *  Always                D$UFRNQDQ
          INCLUDE                D$UFRRLD
          for RMS                D$UFRMEM
                                 D$UFRWFIO

                   (System      (User       (File
                    Calls)       Functions)  Access)
                  |_________________________________|
                       INCLUDE files as needed
```

1. Include if not always using $ERMSG if error occurs.

2. Include if needed.

....continued

```
SYNTAX..: FUNCTIONname (argument_1, anotherArgument,
                         aFunctionArgument,aValueMaybe,
                         oneOrTwoByteValues,
                         aPointerToValue,
                   *     &addressOfVariableName      *
                         upToNineArgumentsEven )
```

NOTE:

The names of arguments will appear in small letters since they represent variables which must be defined by the programmer. Those names must be exclusive to the local or global block of DASL code in which the function statement is used.

The variables must be of the "type" indicated in the
ENTRY n..: or
.. EXIT n: fields of the description.

The ampersand "&" is commonly used in the actual function statements to assign an argument with the value of the "address of" a variable.

```
 --------------------------------------
| * & ... the ADDRESS OF  OPERATOR   * |
|                                      |
| see the discussion that follows in   |
| this section.                        |
|                                      |
|  ALL EXIT PARAMETERS are now shown   |
|     using the Address Of operator.   |
|                                      |
 --------------------------------------
```

.....continued

RESULT..: D$CCODE is a very popular TYPE of result

A program statement of a function name "may" be DEFINEd as a value (not a Variable with a specific memory location).

This resultant value "may" be used in a program statement of some longer expression and/or ASSIGNed to some variable (for example, to do multiple tests on the condition code flags when the function's value is type D$CCODE).

Result values must be of
CLASS: SCALAR or POINTER,
(but not
CLASS: ARRAY, FUNCTION or STRUCTURE/UNION).

D$CCODE is a Type of Value defined (by TYPDEF) as a
Class: SCALAR;
Specific Type: BYTE.

In addition four bits have been defined as:
D$CFLAG,D$ZFLAG,D$SFLAG,D$PFLAG.

```
DASL                     DASL
SCALAR                   POINTER
TYPES:  BYTE             TYPES: (may POINT
        CHAR                     to any TYPE
        BOOLEAN                  in any CLASS )
        INT                      SCALAR
        UNSIGNED                 ARRAY
        LONG                     POINTER
                                 FUNCTION
                                 STRUCTURE/UNION


                         (a POINTER is type
                              UNSIGNED )
```

......continued

```
USE.....: IF $FUNCTION (arg1,arg2) && D$CFLAG
          THEN conditional;

          NOTE:
                Where a conditional statement or label
          would appear in a real statement, in these
          descriptions, the reason for the condition
          is usually stated.

ASSIGN..: varRslt := $FUNCTION (arg1, arg2 );
          IF varRslt && D$ZFLAG
          THEN conditionalStatment;

          NOTE:
                "varRslt" represents a variable to be
          defined.  See the NOTE: in USE.....: above.

ENTRY.1.: argument1 ITS TYPE; and comments
ENTRY.2.: arg2      ITS TYPE; and comments
...
ENTRY.9.: arg9      ITS TYPE; and comments

..EXIT 1: same as ENTRIES
```

```
IF ERROR: IF  FUNCTION (...) && D$CFLAG
          THEN $ERMSG ();
       or IF  varRslt && D$CFLAG THEN $ERMSG ();
       or No Error Occurs
```

For the first 2 cases: possible error code value names which may be in $ERRC.$FUNC and $ERRC.$CODE (variables defined in Include file D$RMS as EXTERNAL type $ERRCODE).

NOTE:
(...) means the argument list defined in SYNTAX.:

```
 ------------------------------------
| NOTE:                              |
| In the FUNCTION Section description|
| of error codes, contents of:       |
|   $ERRC.$FUNC = SC$.... or         |
|                 $UEC...            |
|   $ERRC.$CODE = $EC...nn  or       |
|                 $UEC...nn          |
| The $EC..nn or $UEC..nn ends with a|
| DECIMAL NUMBER.                    |
|       Statements testing the       |
| value of $ERRC.$CODE may use the   |
| word if it is defined in DASL.     |
| Defined values will have :D$fileName|
| or :* (if defined in D$ERRCODE)    |
| following the line of description. |
| If the value is not defined, use the|
| decimal value, not the word.       |
 ------------------------------------
```

SEE ALSO: More info or Functions which are related.

DASL DOC: n, n (pages)   March 1982 or July 1982

SPRM REF: Vol.(n) Sec. (n.n.n.n)   Date: 9-01-82

Use of the *ADDRESS OF* OPERATOR

```
|                                             |
|  As of this Update, ALL EXIT PARAMETERS     |
|  are shown using the ADDRESS OF operator.   |
|_____________________________________________|
```

The ampersand "&" is commonly used in function ;tatements to mean "address of" a variable.

:xample:

Description in DASL DICTIONARY

```
SYNTAX..: $FUNCTION1 (first, second)
ENTRY 1.: first      ^ BYTE;
ENTRY 2.: second     ^ BYTE;
```

Program Usage

```
VAR     DOG, CAT BYTE;
    CATADDRESS ^ BYTE;

   { IF $FUNCTION1 ( &DOG, CATADDRESS ) THEN GO };
or { IF $FUNCTION1 ( &DOG, &CAT ) THEN GO };
```

......are equivalent

All parameters which are exit parameters (and in ;ome cases also entry parameters) may be specified in :he calling sequence SYNTAX with the *ADDRESS OF*)perator (ampersand) because all exit parameters ·equire a pointer to a destination location.

Note that both of the following examples are ;pecifiying the same equivalent function.

In the DASL Dictionary, FUNCTION Descriptions do 1ot show the *ADDRESS OF* operator in the SYNTAX ;xamples. However, all parameters which are exit)arameters have at least one "pointer to" (^) operator in their TYPE specifications.

Therefore, in the calling sequence, you may specify a variable which is of the TYPE specified (e.g. a pointer to a BYTE), or you may specifiy a variable name (preceeded by an ampersand) which is th specified TYPE less one "pointer to " operator (e.g. BYTE).

Note also, if an ampersand preceeds a parameter name in the SYNTAX description, you may choose to write the calling sequence with a variable name not preceeded with an ampersand, but that variable must b a pointer to the TYPE of variable specified.

Restatement of the principle:

If the object is to assign a value to a variabl of some TYPE, and the function parameter requires a pointer to that variable TYPE, then you may specifiy that parameter as the variable name preceeded by the ampersand. Or, you may have an extra predefined variable which is a pointer to the first variable. Yo may then specify the function parameter as the name o the pointer variable (this seems like un-neccessary extra work in many cases).

If the function is specified by one convention in the Dictionary, it will still work if you choose t use it the other way, if that is preferable for some reason.

...example follows

TWO Ways the Functions COULD be Defined

The functions could be defined without specifying *ADDRESS OF* or they could specify *ADDRESS OF*. In either case the net result is the same. Basically putting an ampersand "&" before a variable name in a function calling sequence is equivalent to specifying a value which is a pointer to that variable TYPE.

EXAMPLE of Function Description
without ADDRESS OF **OPERATOR:**

```
SYNTAX..: $WSIO     (mode, string, hor, ver, end)
RESULT..: D$CCODE

ENTRY 1.: mode     ^ BYTE; Initial mode bits ($WSM..)
ENTRY 2.: string   ^ CHAR; Data to be displayed
ENTRY 3.: hor      ^ BYTE; Initial horizontal cur pos
ENTRY 4.: ver      ^ BYTE; Initial vertical cursor pos
EXIT 1..: mode     ^ BYTE; Mode,unless error: invalid
EXIT 3..: hor      ^ BYTE; Position after last char
EXIT 4..: ver      ^ BYTE; Position after last char
EXIT 5..: end    ^ ^ CHAR; Char after string terminator
```

EXAMPLE of Function Description Modified
with ADDRESS OF **OPERATOR:**

```
SYNTAX..: $WSIO     (&mode, string, &hor, &ver, &end)
RESULT..: D$CCODE

ENTRY 1.: mode      BYTE; Initial mode bits ($WSM..)
ENTRY 2.: string  ^ CHAR; Data to be displayed
ENTRY 3.: hor       BYTE; Initial horizontal cursor pos
ENTRY 4.: ver       BYTE; Initial vertical cursor pos
EXIT 1..: mode      BYTE; Mode, unless error: invalid
EXIT 3..: hor       BYTE; Position after last char
EXIT 4..: ver       BYTE; Position after last char
EXIT 5..: end     ^ CHAR; Char after string terminator
```

FUNCTIONS

Explanation of FUNCTION SECTION FORMAT Fields

Please see the more detailed discussion of each field in the preceding section, FORMATS. Also please note the discussion of the use of the ADDRESS OF operator. ***All EXIT parameters* are now shown in the calling syntax examples with *use of the ADDRESS OF operator: &* (ampersand symbol)**

HINT: Always look at the ***TYPE SECTION*** description for the TYPE of each parameter to get more information about those structures, which are not detailed in each FUNCTION description since they are typically used by more than one function.

Title Fields

FUNCTIONNAME	FUNCTION	**FUNCTIONNAME**

Brief Description of Function

General Fields

Category: Function kind (One of ten groups)

Entered : Date added to DASL Dictionary

Updated : Last date when function or comments changed

File : DASL INCLUDE File where Function is Defined

Calling Syntax, Result, and Parameter Fields

Note: In the calling sequence descriptions for each function

1. **BOLD WORDS**: actual words to use
2. ***bold italic WORDS***: in lowercase letters are values or variable name substitutes which represent values or variables to be assigned or declared by the programmer.
3. *Italic WORDS*: following an IF THEN statement represent the the condition meaning, and would actully be replaced in a program with conditional code.

Syntax : Defined Macro Calling Sequence (usually the calling sequence is not used alone but is combined with IF statements: see USE, ASSIGN, and IF ERROR fields.)

Result : Function's resultant value TYPE. Some functions have no defined result. The Result TYPE is not actually written in the program.

Use : Calling syntax to use if only one condition flag test is required, *except for* **TRUE CARRY FLAG**, the standard RMS ERROR condition.

Assign : Calling syntax to use if multiple condition flags are to be tested. First the function is called in a statement assigning its result to a variable.

Note: <u>**Parameters**</u>

1. A DASL Function *Macro* may be defined to have zero to nine parameters in its calling sequence.

2. Parameters may be defined to be either entry parameters or exit parameters, or both.

3. Each parameter will be listed once as an entry parameter or once as an exit parameter or twice if it is both an entry and exit.

4. The number of the parameter in the list of entry and exit parameters, refers to the parameters position in the calling sequence.

5. Parameters will always be 1,2,or 4 byte scalar values or pointers or ULONG structures.

   *NOTE: What we are calling DASL RMS System "Functions", are actually DASL MACROS which have a slightly broader definition than a **FUNCTION** VARIABLE TYPE. A macro is defined with a DEFINE statement, a function is declared in a DASL declaration statement. Macros may have any string as an argument, including any variable, function, macro, etc.*

   Parameters should be specified in a calling sequences as required in one of the following ways:

a) IF a ***Value*** *TYPE* is required, specify:
   NUMERIC VALUE (e.g. 20, 034, 1<<31, 64000
   or ***SINGLE LITERAL VALUE*** (e.g. 'A')
   or ***DEFINED VALUE NAME***
   or ***VARIABLE NAME*** (Automatically assumes value)
   or ***POINTER NAME ^*** (Indirect "variable name")

b) IF a ***Pointer*** *TYPE* is required, specify:
   & VARIABLE NAME (Address of)
   or ***POINTER NAME***

...continued

```
| c) IF the parameter in the Syntax : EXAMPLE       |
|     is of the from &something, specify:           |
|     & VARIABLE NAME if the parameter is a value   |
|  or & POINTER NAME if the parameter is a pointer  |
+---------------------------------------------------+
```

Entry and Exit PARAMETER Fields

Entry 1 : ***variableNameOrValue*** TYPE; Comments
thru
Entry 9 : ***variableNameOrValue*** TYPE; Comments
* *Exit 1*: ***variableName*** TYPE; Comments
thru
* *Exit 9*: ***variableName*** TYPE; Comments

Standardized ERROR Calling Syntax Field

If Error: Calling syntax to use if the function has a possible RMS error condition. This will always be a test of the TRUE CARRY FLAG condition.

A Call to $ERMSG () will be shown if that could be used. The programmer may opt to make tests on the $ERRC bytes to decide to execute some other code, rather than abort through $ERMSG. Possible contents of $ERRC will be listed.

Often this is the complete statement form required to use the function, unless an ASSIGN statement, or group of statements is specified, in which case the IF ERROR statement is used in sequence with them.

Miscellaneous Fields

Remarks : Miscellaneous Information

See Also: Cross Referencing and Association of Functions

DASL Doc: The DASL Document, March 1982 (page reference)

SPRM Ref: RMS System Programmer's Ref Manual
Volume and Section References
*** Update 1 September 1982 ***

$ACLOSE FUNCTION $ACLOSE

Close an AIM File

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$FAR

Syntax  : $ACLOSE   (work, fcb, mode )
Result  : D$CCODE

Entry 1 : work   ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb    ^ $FCBA; FCB
Entry 3 : mode     BYTE; Close mode ($CM...)

If Error: IF  $ACLOSE (...) && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS4,$ECFMS5,
                        $ECFMS6
SPRM Ref: $ACLOSE       Vol.3 Sec.  5.3.1.2
```

$ADELCR FUNCTION $ADELCR

AIM Delete Current Record Read by $AREAD,$AREADKG

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$FAR

Syntax  : $ADELCR   (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $ADELCR (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb   ^ $FCBA; FCB
* Exit 3: err     BYTE; Exception code ($XCFMS06)

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS54
SPRM Ref: $ADELCR         Vol.3 Sec.  5.3.2.3
```

$AINS FUNCTION $AINS

Insert Key into AIM Index for Existing Data Recd

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$FAR

Syntax  : $AINS     (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $AINS (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb   ^ $FCBA; FCB
* Exit 3: err     BYTE; Exception codes (none in SPRM)?

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS50,$ECFMS
SPRM Ref: $AINS           Vol.3 Sec.  5.3.3.3
```

Complete Pending Writes,Force Buffer fill on Reads

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$FAR

Syntax : **$AIOCLR** ***(work, fcb)***
Result : *D$CCODE*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBA;** FCB

If Error: **IF $AIOCLR (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $AIOCLR Vol.3 Sec. 5.3.1.3

Initialize AIM File Access

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$FAR

Syntax : **$AOPEN** ***(work, fcb, mode, openpt, name)***
Result : *D$CCODE*

Entry 1 :	***work***	**^** [256] **BYTE**; Paged buffer, D$BUFn
Entry 2 :	***fcb***	**^ $FCBA**; FCB
Entry 3 :	***mode***	**BYTE**; Open mode ($OM...)
Entry 4 :	***openpt***	**^ $OPENPT**; Open parameter table
Entry 5 :	***name***	**^ $NAMEEXTENV**; File Name Ext Env

If Error: **IF $AOPEN (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS1,$ECFMS2,$ECFMS3,
$ECFMS4,$ECFMS5,$ECFMS6,$ECFMS7,$ECFMS11,
$ECFMS19,$ECFMS20,$ECFMS21,$ECFMS46,
$ECFMS47,$ECFMS48,$ECFMS49,$ECFMS59
Remarks : $AOPEN must be called before any FMS calls
to this $FCBA.
See Also: $OPENENV for Open Modes, Access Codes,
Formats
SPRM Ref: $AOPEN Vol.3 Sec. 5.3.1.1

Position to Logical Record Meeting Key List Spec

```
Category: File Access Routine
Entered : 82 Jul 01              Updated : 83 Jul 01
File    : D$FAR

Syntax  : $APOS      (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $APOS (work, fcb, &err):
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb   ^ $FCBA; FCB
* Exit 3: err     BYTE; Exception code ($XCFMS02)

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS50,
                $ECFMS51,$ECFMS52,$ECFMS55
Remarks : Subsequent $AREADKG operations access data.
SPRM Ref: $APOS         Vol.3 Sec.  5.3.3.4
```

AIM Read Logical Record Meeting Key List Spec

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$FAR

```
Syntax   : $AREAD    (work, fcb, &err, &end)
Result   : D$CCODE
Assign   : varRslt := $AREAD (work, fcb, &err, &end);
           IF varRslt && D$ZFLAG
           THEN ("err" has exception code);

Entry 1  : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2  : fcb   ^ $FCBA; FCB
* Exit 3 : err     BYTE; Exception code ($XCFMS02)
* Exit 4 : end   ^ CHAR; Last char stored in user area

If Error : IF  varRslt && D$CFLAG THEN $ERMSG ( );
           $ERRC.$FUNC = $FMS
           $ERRC.$CODE = see $APOS for error codes
SPRM Ref : $AREAD          Vol.3 Sec.  5.3.3.1
```

Read after $APOS or Re-read Current after $AREAD

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$FAR

Syntax  : $AREADCR  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $AREADCR  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb   ^ $FCBA; FCB
* Exit 3: err     BYTE; Exception codes ($XCFMS06)
* Exit 4: end   ^ CHAR; Last char stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $AREADCR       Vol.3 Sec.  5.3.2.2
```

AIM Read Key Generic, Read Records with Same Key

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$FAR

Syntax : **$AREADKG** ***(work, fcb, &err, &end)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $AREADKG *(work, fcb, &err, &end);***
IF *varRslt* && D$ZFLAG
THEN *("err" has exception code);*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBA;** FCB
** Exit 3*: ***err*** **BYTE;** exception codes ($XCFMS01,$XCFMS
** Exit 4*: ***end*** **^ CHAR;** Last char stored in user area

If Error: **IF *varRslt* && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS53
Remarks : Use after $APOS to read that record. Use to read subsequent records then or after $AREAD.
SPRM Ref: $AREADKG Vol.3 Sec. 5.3.2.1

AIM Rewrite Current Record after $AREAD, $AREADKG

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$FAR

Syntax  : $ARWRTCR  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $ARWRTCR  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb   ^ $FCBA; FCB
* Exit 3: err     BYTE; Exception code ($XCFMS06)
* Exit 4: end   ^ CHAR; Last char read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS54
SPRM Ref: $ARWRTCR      Vol.3 Sec.  5.3.2.4
```

Write Recd at End of Data File,Insert Key in Index

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$FAR

Syntax  : $AWRITE   (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $AWRITE   (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work   ^ [256] BYTE; Paged buffer, D$BUFn
Entry 2 : fcb    ^ $FCBA; FCB
* Exit 3: err      BYTE; Exception codes ($XCFMS...)
* Exit 4: end    ^ CHAR; Last char read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS50
SPRM Ref: $AWRITE        Vol.3 Sec.  5.3.3.2
```

$BASESET FUNCTION $BASESET

Set the Memory Base Register

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$RMSMEM

Syntax : **$BASESET** ***(base, &old)***
Result : *D$CCODE*

Entry 1 : ***base*** BYTE; New base value
** Exit 2*: ***old*** BYTE; Previous base value

If Error: **No Error Occurs**
SPRM Ref: $BASESET Vol.4 Sec. 5.2.5

$BEEP FUNCTION $BEEP

Workstation, Make a Beep Sound

Category: System Call
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMSWS

Syntax : **$BEEP** *()*
Result : *D$CCODE*

If Error: **IF $BEEP (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$WSCTL
$ERRC.$CODE = $ECWSCC1
See Also: Uses System Call $WSCTL
SPRM Ref: $BEEP Vol.4 Sec. 11.9

BIGBCP$ FUNCTION BIGBCP$

Compare Large Strings, ASM Language UFR

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : SEE ***D$COMP, DASL EXTERNAL FUNCTION equivalent***
SPRM Ref: BIGBCP$ Vol.2 Sec. 8.3

BIGBT$ FUNCTION BIGBT$

Move 16-bit Length, ASM Language UFR

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : SEE ***D$MOVE, DASL EXTERNAL FUNCTION equivalent***
SPRM Ref: BIGBT$ Vol.2 Sec. 8.1

$BINOC24 FUNCTION $BINOC24

Numeric, Convert 24-Bit Binary to ASCII Octal

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRNUM

Syntax : **$BINOC24** ***(value, right, size)***
Result : *none*

Entry 1 :	***value***		**ULONG**; 24 bit value to convert
Entry 2 :	***right***	^	**CHAR**; Output area end:low order byte
Entry 3 :	***size***		**BYTE**; Output bytes desired

If Error: **No Error Occurs**
SPRM Ref: BINOC24$ Vol.2 Sec. 4.1

Numeric, Convert 16-bit Binary to ASCII Octal

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRNUM

Syntax : **$BINOCT** ***(value, right, size)***
Result : *none*

Entry 1 :	***value***	**UNSIGNED;** 16 bit value to convert
Entry 2 :	***right***	**^ CHAR;** Output area end:low order byte
Entry 3 :	***size***	**BYTE;** Output bytes desired: 0 is legal

If Error: **No Error Occurs**
SPRM Ref: BINOCT$ Vol.2 Sec. 4.2

CASE FUNCTION CASE

Transfer Control to Statement Label equal to Argument

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : **CASE *arg {label1:stmnt; label2:stmnt; label3:stmnt;*....DEFAULT:*stmnt;}***;

Remarks : Labels may be defined numeric or character values consistent with the argument type used. The order of listing labels is not important, except DEFAUL must be last.
DASL Doc: 27,74 March 1982

$CHAININ FUNCTION $CHAININ

Workstation-IF, Determine if CHAINing is Active

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWS

Syntax : **$CHAININ** *()*
Result : *D$CCODE*
Use : **IF $CHAININ** *()* **&& D$CFLAG=0 THEN** *(CHAIN not Active)*

If Error: **No Error Occurs**
SPRM Ref: CHAININ$ Vol.2 Sec. 7.20

Workstation, make a click sound

Category: System Call
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMSWS

Syntax : **$CLICK** *()*
Result : *D$CCODE*

If Error: **IF $CLICK () && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$WSCTL
$ERRC.$CODE = $ECWSCC1
SPRM Ref: $CLICK Vol.4 Sec. 11.8

Close a File

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25
File    : D$RMS

Syntax  : $CLOSE    (mode, pfdb)
Result  : D$CCODE

Entry 1 : mode  BYTE; close modes ($CM...)
Entry 2 : pfdb  ^ $PFDB;

If Error: IF $CLOSE(...) && D$CFLAG THEN $ERMSG ();   |*
          $ERRC.$FUNC = SC$CLOSE
          $ERRC.$CODE = All resources                  |*
                $ECUMAV,$ECSI001,$ECSI005,$ECSI007,    |*
                $ECSI010,$ECSI034                      |*

                        Disk resources                 |*
                $ECSI012,$ECSI013,$ECSI014,$ECSI015,   |*
                $ECSI021,$ECSI032,$ECSI045             |*

                        Pipe resources                 |*
                $ECSI021,$ECSI045                      |*

                        Printer resources              |*
                $ECSI021,$ECSI045                      |*

Remarks : Close Modes: $CMUNCH,$CMCHOP,$CMSIZE,$CMKILL
          $CLOSE does not flush buffers. $SECWAIT or
          other flush operations must complete first.
SPRM Ref: $CLOSE        Vol.4 Sec.  7.4.3
SPRM Ref: $CLOSE        Vol.4 Sec.  8.4.5
SPRM Ref: $CLOSE        Vol.4 Sec.  9.1.1.5
SPRM Ref: $CLOSE        Vol.4 Sec.  9.2.1.3
SPRM Ref: $CLOSE        Vol.4 Sec.  9.3.1.3
SPRM Ref: $CLOSE        Vol.4 Sec.  9.4.1.3
SPRM Ref: $CLOSE        Vol.4 Sec. 10.1.3
```

$CLOSEAL FUNCTION $CLOSEAL

Close All Open Files

Category: System Call
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMSIO

Syntax : **$CLOSEAL** *()*
Result : *D$CCODE*

If Error: **No Error Occurs**
Remarks : Performs a $CLOSE in $CMUNCH mode on all FAV's owned by the Task.
See Also: $$CLOSEAL for CHAIN or LOG compatibility
SPRM Ref: $CLOSEAL Vol.4 Sec. 7.4.4

$$CLOSEAL FUNCTION $$CLOSEAL

Workstation-IF, Interface to $CLOSEAL System Call

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWS

Syntax : **$$CLOSEAL** *()*
Result : *D$CCODE*

If Error: **No Error Occurs**
Remarks : Use instead of System Call to guarantee CHAIN and LOG compatibility.
See Also: $CLOSEAL
SPRM Ref: CLOSEAL$ Vol.2 Sec. 7.16

$CONDEC FUNCTION $CONDEC

Numeric, Convert ASCII Decimal String to Binary

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRNUM

Syntax : **$CONDEC** ***(&scanpt)***
Result : *UNSIGNED*

Entry 1 : **scanpt** **^ CHAR;** Start of decimal string
** Exit 1*: **scanpt** **^ CHAR;** Non-numeric terminator
or last byte converted + 1
If Error: **No Error Occurs**
SPRM Ref: CONDEC$ Vol.2 Sec. 4.3

$CONOC24 FUNCTION $CONOC24

Numeric, Convert ASCII Octal to 24-Bit Binary

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRNUM

Syntax : **$CONOC24** ***(&scanpt, &value)***
Result : *none*

Entry 1 : **scanpt** **^ CHAR;** Start of ASCII string
** Exit 1*: **scanpt** **^ CHAR;** Terminator: non-(0 thru 7)
** Exit 2*: **value** **ULONG;** 24-bit binary value

If Error: **No Error Occurs**
SPRM Ref: CONOC24$ Vol.2 Sec. 4.4

Numeric, Convert ASCII Octal to 16-Bit Binary

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRNUM

Syntax : **$CONOCT** ***(&scanpt)***
Result : *UNSIGNED*

Entry 1 : ***scanpt*** **^ CHAR;** Start of ASCII string
** Exit 1*: ***scanpt*** **^ CHAR;** Terminator: non-(0 thru 7)

If Error: **No Error Occurs**
SPR*M Ref*: CONOCT$ Vol.2 Sec. 4.5

$CSCPUSH FUNCTION $CSCPUSH

Workstation, Push Command Lines Below Current Pointer

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWS

Syntax : $CSCPUSH *(lines, length)*
Result : *D$CCODE*

Entry 1 : ***lines*** **^ CHAR**; strings terminated by $WSENTK
Entry 2 : ***length*** **UNSIGNED**; length of set of strings

If Error: **IF $CSCPUSH (...) && D$CFLAG**
THEN (won't fit on stack)
SPRM Ref: CSCPUSH$ Vol.2 Sec. 7.14

$CSPOP FUNCTION $CSPOP

Workstation-IF, Pop The Command Stack

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWS

Syntax : **$CSPOP** *()*
Result : *D$CCODE*
Use : **IF $CSPOP *()* && D$ZFLAG**
THEN (command stack empty)

If Error: **No Error Occurs**
SPRM Ref: CSPOP$ Vol.2 Sec. 7.12

$CSPUSH FUNCTION $CSPUSH

Workstation,Push a Command Line Onto Command Stack

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$UFRWS

Syntax  : $CSPUSH   (line)
Result  : D$CCODE

Entry 1 : line  ^ CHAR; string terminated by $WSENTK

If Error: IF  $CSPUSH (...) && D$CFLAG
          THEN (won't fit on stack)
SPRM Ref: CSPUSH$       Vol.2 Sec.  7.13
```

$CSPUSHN FUNCTION $CSPUSHN

Workstation,Push Command Lines Onto Command Stack

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Apr 23
File    : D$UFRWS

Syntax  : $CSPUSHN  (lines, length)
Result  : D$CCODE

Entry 1 : lines  ^ CHAR;
Entry 2 : length  UNSIGNED;

If Error: IF  $CSPUSHN (...) && D$CFLAG
          THEN (won't fit on stack)

Remarks : allows a series of strings to be pushed on
          the stack at current pointer position
SPRM Ref: CSPUSHN$      Vol.2 Sec.  7.15
```

$CURSOFF FUNCTION $CURSOFF

Workstation, Turn Off the Cursor

Category: System Call
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMSWS

Syntax : **$CURSOFF** *()*
Result : *D$CCODE*

If Error: **IF $CURSOFF (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$WSCTL
$ERRC.$CODE = $ECWSCC1
Remarks : Uses SYSTEM CALL $WSCTL
SPRM Ref: $CURSOFF Vol.4 Sec. 11.7

$CURSON FUNCTION $CURSON

Workstation, Turn On the Cursor

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$RMSWS

Syntax : **$CURSON** *(hor, ver)*
Result : *D$CCODE*

Entry 1 : ***hor*** **BYTE;** horizontal position ($WSRC,$WSLC)
Entry 2 : ***ver*** **BYTE;** vertical position ($WSBL)

If Error: **IF $CURSON (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$WSCTL
$ERRC.$CODE = $ECUMAV,$ECWSCC1,$ECWSCC2

Remarks : Uses SYSTEM CALL $WSCTL...cursor stays on unti
turned off or $WSIO function writes to screen
See Also: $WCONFIG for screen size.
SPRM Ref: $CURSON Vol.4 Sec. 11.6

Numeric, Convert ASCII Decimal to 16-Bit Binary

Category:	User Function Routine		
Entered :	82 Jul 01	*Updated* :	83 Apr 02
File :	D$UFRNUM		

Syntax :	**$CVB**	***(&scanpt, size)***
Result :	*UNSIGNED*	

Entry 1 :	***scanpt***	**^ CHAR**; Left-most byte of string
Entry 2 :	***size***	**BYTE**; Number of bytes to convert
** Exit 1*:	***scanpt***	**^ CHAR**; Left-most plus "size"

If Error:	**No Error Occurs**	
See Also:	Same as $CONDEC except has no terminator.	
SPRM Ref:	CVB$	Vol.2 Sec. 4.6

Numeric, Convert ASCII Decimal to 24-Bit Binary

Category:	User Function Routine		
Entered :	82 Jul 01	*Updated* :	83 Apr 02
File :	D$UFRNUM		

Syntax :	**$CVB24**	***(&scanpt, &value)***
Result :	*none*	

Entry 1 :	***scanpt***	**^ CHAR**; Left-most ASCII byte
** Exit 1*:	***scanpt***	**^ CHAR**; Terminator: non-numeric
** Exit 2*:	***value***	**ULONG**; 24-bit binary value

If Error:	**No Error Occurs**	
SPRM Ref:	CVB24$	Vol.2 Sec. 4.7

$CVB24L FUNCTION $CVB24L

Numeric, Convert ASCII Decimal to 24-Bit Binary

Category:	User Function Routine		
Entered :	82 Jul 01	*Updated* :	83 Apr 02
File :	D$UFRNUM		

Syntax : **$CVB24L** ***(&scanpt, size, &value)***
Result : *none*

Entry 1 :	***scanpt***	^ **CHAR**; Left-most ASCII byte
Entry 2 :	***size***	**BYTE**; Number of bytes to convert
* *Exit 1*:	***scanpt***	^ **CHAR**; Left-most plus "size"
* *Exit 3*:	***value***	**ULONG**; 24-bit value

If Error: **No Error Occurs**
See Also: Same as $CVB24 except has no terminator.
SPRM Ref: CVB24L$ Vol.2 Sec. 4.8

$CVD FUNCTION $CVD

Numeric, Convert 16-Bit Binary to ASCII Decimal

Category:	User Function Routine		
Entered :	82 Jul 01	*Updated* :	82 Dec 01
File :	D$UFRNUM		

Syntax : **$CVD** ***(value, left, size)***
Result : *none*

Entry 1 :	***value***	**UNSIGNED**; Binary value to convert
Entry 2 :	***left***	^ **CHAR**; Left-most output byte
Entry 3 :	***size***	**BYTE**; Output length (1 to 5 bytes)

If Error: **No Error Occurs**
See Also: $CVDZ for zero-suppression other than blank
SPRM Ref: CVD$ Vol.2 Sec. 4.9

Numeric, Convert 24-Bit Binary to Decimal

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRNUM

Syntax : **$CVD24** ***(value, left, size)***
Result : *none*

Entry 1 :	***value***		**ULONG;** Binary value to convert
Entry 2 :	***left***	**^**	**CHAR;** Left-most output byte
Entry 3 :	***size***		**BYTE;** Output area (1 to 8 bytes)

If Error: **No Error Occurs**
See Also: $CVD24Z for zero-suppression other than blank
SPRM Ref: CVD24$ Vol.2 Sec. 4.10

Numeric, $CVD24 with Zero-Suppression Non-Blank

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRNUM

Syntax : **$CVD24Z** ***(value, left, size, fill)***
Result : *none*

Entry 1 :	***value***		**ULONG;** Binary value to convert
Entry 2 :	***left***	**^**	**CHAR;** Left-most output area
Entry 3 :	***size***		**BYTE;** Output field length (1-8)
Entry 4 :	***fill***		**CHAR;** Fill character

If Error: **No Error Occurs**
See Also: $CVD24 for zero suppression using blank
SPRM Ref: Vol.2 Sec. 4.10

Numeric, $CVD with Zero-Suppression Non-Blank

Category:	User Function Routine	
Entered :	82 Jul 01	*Updated* : 83 Jul 10
File :	D$UFRNUM	

Syntax :	**$CVDZ**	***(value, left, size, fill)***
Result :	*none*	

Entry 1 :	***value***		**UNSIGNED**; Binary value to convert
Entry 2 :	***left***	^	**CHAR**; Left-most output byte
Entry 3 :	***size***		**BYTE**; Output field (1 to 5 bytes)
Entry 4 :	***fill***		**CHAR**; Zero-suppression character

If Error:	**No Error Occurs**	
See Also:	$CVD for zero suppression using blank	
SPRM Ref:	CVD$	Vol.2 Sec. 4.9

General-Utility, Obtain System Date and Time Info

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRGEN

Syntax : **$CVSTIME** ***(time, output)***
Result : *none*

Entry 1 : ***time*** **^ $TIME;** Seconds since 01/01/1901
Entry 2 : ***output*** **^ $CVSTBL;** month,day,year,time,etc.

If Error: **No Error Occurs**
SPRM Ref: CVSTIME$ Vol.2 Sec. 8.6

Call an Arbitrary Externally Defined Subroutine

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$INC

Syntax : **D$CALL** *(Function)*
Result : *D$CCODE*

Entry 1 : ***Function*** **^ D$CALLF;** Pointer to subroutine

If Error: **"usually" IF D$CCODE && D$CFLAG**
THEN $ERMSG () or other action

Remarks :

Calls an external routine which has been declared TYPE D$CALLF. D$CALL is used in the DEFINE statements for FAR and UFR external routine macros. (System Calls use D$SC).

Like DSC, DCALL requires the macro DEFINE statement to first assign argument values
e.g. FUNCTION (argument,argument,..)
into the Processor Register Variables:
DX,DA,DB,DC,DD,DE,DH,DL,
DXA,DBC,DDE,DHL
(for 5500,6600 processor series).

The D$CALL function then calls the external function which is its argument. On return, the D$CALL routine places the processor register values back into the Processor Register Variables,and puts the processor flag conditions in the D$CCODE type byte D$CC.

The DEFINE statement then assigns any pertinent register values into the macro function's arguments.

$ERRC is defined as EXTERNAL type $ERRCODE and always contains the contents of D$BC for use with RMS error message coding and checking.

EXAMPLE:
Macro Statement (this is what the user uses):

```
    $CVB24L  (scanpt, size, value)
```

External Function Declaration (in Include File):

```
    EXTERN CVB24L$ D$CALLF; /* the UFR routine */
```

Macro Definition (in Include File):

```
    DEFINE($CVB24L, {         /* Define the macro */
       <^CHAR>D$HL:=(#1)^;   /* scanpnt in HL    */
       D$A:=(#2);            /* size in A        */
       D$CALL(&CVB24L$);     /* call             */
       (#1)^:=<^CHAR>D$HL;   /* HL to scanpt     */
       (#3)^:=<ULONG>D$E;    /* E to value       */
                      } )
```

NOTE: The Processor Register Variables should not be referenced by USER programs except to DEFINE interfaces to user defined external routines, otherwise that program is processor series dependent.

See Also: D$SC for SYSTEM CALL External Function
DASL Doc: 90-91 March 1982

Block Compare Two Character Strings

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$INC

Syntax : **D$COMP** ***(source, dest, num)***
Result : *INT*
Zero if the two strings are equal,
else Difference of first unmatched bytes:
SourceByte minus DestinationByte

Entry 1 : ***source*** **^ BYTE;** Character string 1
Entry 2 : ***dest*** **^ BYTE;** Character string 2
Entry 3 : ***num*** **UNSIGNED;** Number of bytes to compare

If Error: **No Error Occurs**
Remarks : Replaces UFR BIGBCP$
DASL Doc: 90 March 1982
SPRM Ref: BIGBCP$ Vol.2 Sec. 8.3

D$GET24 FUNCTION **D$GET24**

Numeric, Convert 24 Bit to 32 Bit Value

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$INC

Syntax : **D$GET24** ***(pul)***
Result : *LONG*

Entry 1 : ***pul*** **^ ULONG ;** 24 bit value

If Error: **No Error Occurs**
DASL Doc: 90-98 12 JUL 1982

D$INFO FUNCTION D$INFO

Return Processor Type

```
Category: DASL External Function
Entered : 83 Apr 02                  Updated : 84 Jul 01   |*
File    : D$INC

Syntax  : D$INFO   ()
Result  : BYTE

If Error: No Error Occurs
REMARKS.: The result value for different processor      |*
          types is:                                     |*

           5000 series: 0                               |*
           6000 series: 1                               |*
    1800 & 3800 series: 2                               |*
           8800 series: 3                               |*
           8600 series: 5                               |*
           8400 series: 9                               |*
```

D$JUMP FUNCTION D$JUMP

Jump to an Arbitrary Externally Defined Subroutine

```
Category: DASL External Function
Entered : 82 Dec 01                  Updated : 84 Aug 8    |*
File    : D$RMS                                            |*

Syntax  : D$JUMP   (function)
Result  : Does not return

Entry 1 : function  ^ D$CALLF; Function to jump to

If Error: No Error Occurs
```

D$MOVE FUNCTION D$MOVE

Block Move 0 to 65535 Bytes

Category:	DASL External Function
Entered :	82 Jul 01 *Updated* : 82 Dec 01
File :	D$INC
Syntax :	**D$MOVE** ***(source, dest, num)***
Result :	*none*
Entry 1 :	***source*** **^ BYTE;** Address of bytes to be moved
Entry 2 :	***dest*** **^ BYTE;** Address to be moved to
Entry 3 :	***num*** **UNSIGNED;** Number of bytes to move
If Error:	**No Error Occurs**
Remarks :	Replaces UFR BIGBT$
DASL Doc:	90 March 1982

D$MOVER FUNCTION D$MOVER

Block Move Reverse, Starting at Ending Address

Category:	DASL External Function
Entered :	82 Jul 01 *Updated* : 82 Dec 01
File :	D$INC
Syntax :	**D$MOVER** ***(source, dest, num)***
Result :	*none*
Entry 1 :	***source*** **^ BYTE;** Address of bytes to be moved
Entry 2 :	***dest*** **^ BYTE;** Address to be moved to
Entry 3 :	***num*** **UNSIGNED;** Number of bytes to move
If Error:	**No Error Occurs**
Remarks :	Similar to D$MOVE except decrements address from end of memory segment to beginning.
DASL Doc:	90-98 12 JUL 1982

Numeric, Convert 32 Bit to 24 Bit Value

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 84 Jul 01 |*
File : D$INC

Syntax : **D$PUT24** ***(l, &pul)***
Result : *none*

Entry 1 :	*l*	**LONG;** 32 bit value
* *Exit 2*:	***pul***	**ULONG;** 24 bit value |*

If Error: **No Error Occurs**
DASL Doc: 90-98 12 JUL 1982

Call a SYSTEM CALL External Function

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$INC

Syntax : **D$SC** ***(scnum)***
Result : *D$CCODE*

Entry 1 : ***scnum*** **BYTE;**

If Error: **IF D$SC (...) && D$CFLAG THEN $ERMSG ();**
Remarks :
This externally defined DASL macro function will not be used by user programs. It is described for the purpose of understanding the DASL include files.

D$SC is identical to D$CALL except that the argument of D$SC is a number (e.g. 1 thru 60) which is the defined number of a particular System Call, whereas D$CALL has an argument which is a pointer to a function type D$CALLF.

Secondly, D$SC executes an SC (system call) to a non-user mode routine, where as D$CALL executes a normal CALL instruction in user mode.

Otherwise the two functions use the Processor Register Variables and the D$CCODE byte (D$CC) in the same manner.

See Also: D$CALL for calling externally defined subroutin
DASL Doc: 90 March 1982

$DATETIM FUNCTION $DATETIM

Convert System Time to Standard

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$UFRGEN

Syntax : **$DATETIM** ***(time, months, &string)***
Result : *BYTE* (string length)

Entry 1 : ***time*** **^ $TIME;** System time, 5-byte
Entry 2 : ***months*** **^ [12] [3] CHAR;** Month-Name table
** Exit 3*: ***string*** **^ CHAR;** DD MMM YYYY HH:MM

If Error: **No Error Occurs**
Remarks : Month name table previously read from utility message member or initialized by program.

See Also: $GDATTIM. Both use CVSTIME$ as subroutine.
SPRM Ref: DATETIM$ Vol.2 Sec. 8.7

$DCLOSE FUNCTION $DCLOSE

Direct Access, Close

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$FAR

Syntax : **$DCLOSE** ***(work, fcb, mode)***
Result : *D$CCODE*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBD;** Link to file being closed
Entry 3 : ***mode*** BYTE; Close mode ($CM...)

If Error: **IF $DCLOSE (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS4,$ECFMS5
Remarks : Flushes buffers if needed.
SPRM Ref: $DCLOSE Vol.3 Sec. 3.3.1.2

Direct Random Access, Delete

```
Category: File Access Routine
Entered : 82 Jul 01               Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DDEL    (work, fcb, fp, &err)
Result  : D$CCODE
Assign  : varRslt := $DDEL (work, fcb, fp, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; $XCFMS01,$XCFMS02

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS36
Remarks : Fills record with $LDEL characters
SPRM Ref: $DDEL         Vol.3 Sec.  3.3.3.3
```

$DDELCR FUNCTION $DDELCR

Direct Sequential Access, Delete Current

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DDELCR   (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $DDELCR  (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
* Exit 3: err     BYTE; $XCFMS02, $XCFMS06

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS36
SPRM Ref: $DDELCR       Vol.3 Sec.  3.3.2.7
```

$DECOHSI FUNCTION $DECOHSI

Environment, Decompress an HSI String

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$UFRENV

Syntax  : $DECOHSI  (source, dest, &dend)
Result  : D$CCODE

Entry 1 : source  ^ CHAR;
Entry 2 : dest    ^ CHAR;
* Exit 3: dend    ^ CHAR;

If Error: IF $DECOHSI (...) && D$ZFLAG
          THEN HSI format invalid
SPRM Ref: DECOHSI$      Vol.2 Sec.  2.13
```

DEFAULT FUNCTION DEFAULT

Preceeds Statement in CASE for No-match Condition

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **see CASE**
DASL Doc: 74 March 1982

DEFINE FUNCTION DEFINE

Define a String to be Substituted for Identifier

Category: DASL Compiler Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : **DEFINE** ***(identifier,definitionString)***
Result : *null string*
plus substitution wherever identifier appears
plus an optional value result
Remarks :

```
Evaluation suppression marks:    #[ string #]
Parameter definitions:           #1, #2,...#9
are used in DEFINE and other macro statements.
See DASL DOCUMENT; section 8. Macros
```

Assigning a value to a Macro function (like all of these RMS System Function interfaces) is done in a DEFINE statement.

If the last argument in the DEFINE statement is NOT an assignment, but an expression (like a variable name or parameter), that value will be assigned to the Macro as a "resultant" value.

DO NOT CONFUSE functions (which are a variable type with up to 13 parameters) and Macros which define functions (with up to 9 parameters). BOTH are used in statements as callable functions. BOTH may have resultant values. Macros, however are used in other applications besides defining functions.

Remember that the word "function" basically means a verb; something that does something. We use the word function to mean:

1. a DASL variable type, possibly combining DEFINED Macro calls with more code.

2. an RMS System Call, UFR or FAR routine that has been interfaced with a DASL DEFINE definition.

3. other uses of DEFINE to do things, like ENUM, SET, etc.

4. DASL compiler defined words are listed in the FUNCTIONS section of this dictionary because they in fact have a "function" in the compile process.

DASL Doc: 21,29-31,78-80 March 1982

$DGETCRK FUNCTION $DGETCRK

Direct Sequential Access, Get Current Record Key

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DGETCRK  (work, fcb, fp, &err)
Result  : D$CCODE
Assign  : varRslt := $DGETCRK (work, fcb, fp, &err);
          IF varRslt && D$ZFLAG
          THEN ("err" has exception code);

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; exception code $XCFMS06

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DGETCRK      Vol.3 Sec.  3.3.2.2
```

$DGETNRK FUNCTION $DGETNRK

Direct Sequential Access, Get Next Record Key

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$FAR

Syntax  : $DGETNRK  (work, fcb, fp)
Result  : D$CCODE

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;

If Error: IF  $DGETNRK (...) && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DGETNRK      Vol.3 Sec.  3.3.2.1
```

$DIOCLR FUNCTION $DIOCLR

Direct Access, I/O Clear

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$FAR

Syntax  : $DIOCLR   (work, fcb)
Result  : D$CCODE

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBD;

If Error: IF  $DIOCLR (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DIOCLR       Vol.3 Sec.  3.3.1.3
```

$DISCONT FUNCTION $DISCONT

Disconnect from Remote Node |*

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $DISCONT  (envData)
Result  : D$CCODE

Entry 1 : envData  ^ CHAR; Names of net and node to     |*
                           disconnect                   |*

If Error: IF  $DISCONT (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$DISCONT                      |*
          $ERRC.$CODE = $ECUMAV,$ECSI016,$ECSI017,      |*
                        $ECSI021,$ECSI045,$ECSI050      |*
Remarks : Request will not be honored if any files are |*
          open between nodes.                           |*
```

$DISORT FUNCTION $DISORT

Diminishing Increment In-Core Sort,(Ascending)

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRGEN

Syntax : **$DISORT** ***(param)***
Result : *none*

Entry 1 : ***param*** **^ $DISTBL;** Table of fixed-length entries with fixed-length keys

If Error: **No Error Occurs**
SPRM Ref: DISORT$ Vol.2 Sec. 8.5

$DISPCH FUNCTION $DISPCH

Workstation-IF, Display One Character

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS

Syntax : **$DISPCH** ***(hor, ver, char)***
Result : *D$CCODE*

Entry 1 : ***hor*** BYTE; $WSLC ...
Entry 2 : ***ver*** BYTE; $WSBL to ... (depends on WS type)
Entry 3 : ***char*** CHAR;

If Error: **IF $DISPCH (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$WSIO
$ERRC.$CODE = $EC...
Remarks : overhead of single character operations should be avoided
See Also: $WSIO error codes. $WCONFIG for screen size.
SPRM Ref: DISPCH$ Vol.2 Sec. 7.9

$DIVID3 FUNCTION $DIVID3

Numeric, Unsigned 24-bit Division

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRNUM

Syntax : **$DIVID3** ***(dvisor,dividend,&quotient,&remain)***
Result : *D$CCODE*
Use : **IF $DIVID3** ***(...)*** **&& D$ZFLAG**
THEN *(division by zero attempted);*

Entry 1 :	***dvisor***	**ULONG;** Divisor
Entry 2 :	***dividend***	**ULONG;** Dividend
** Exit 3*:	***quotient***	**ULONG;** Quotient
** Exit 4*:	***remain***	**ULONG;** Remainder

If Error: **No Error Occurs**
SPRM Ref: DIVID3$ Vol.2 Sec. 4.15

$DLMCHEK FUNCTION $DLMCHEK

Workstation-IF, Check Character For a Delimiter

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRWS

Syntax : **$DLMCHEK** ***(char)***
Result : *D$CCODE*
Use : **IF $DLMCHEK** ***(char)*** **&& D$CFLAG**
THEN *("char" is a delimiter);*

Entry 1 : ***char*** **CHAR;**

If Error: **No Error Occurs**
Remarks : Checks character against standard delimiter table coded in nucleus
SPRM Ref: DLMCHEK$ Vol.2 Sec. 7.10

Donate a FAV to a Specified Task

```
Category: System Call
Entered : 82 Dec 01                  Updated : 84 Jul 25 |*
File    : D$RMSSPEC

Syntax  : $DONATFV  (pfdb, name)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : name  ^ $NAMET;

If Error: IF  $DONATFV (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$DONATV                    |*
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI010,   |*
                        $ECDFV2                      |*
```

Direct Access, Open

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$FAR

Syntax : $DOPEN ***(work, fcb, mode, openpt)***
Result : *D$CCODE*

Entry 1 : ***work*** ^ **[256] BYTE**; Paged buffer, D$BUFn
Entry 2 : ***fcb*** ^ **$FCBD**; address of FCB
Entry 3 : ***mode*** BYTE; open mode, $OM...(see $OPENENV)
Entry 4 : ***openpt*** ^ **$OPENPT;**

If Error: **IF $DOPEN (...) && D$CFLAG THEN $ERMSG ( );**

```
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS1,$ECFMS2,$ECFMS3,$ECFMS4,
                $ECFMS5,$ECFMS6,$ECFMS7,$ECFMS21,
                $ECFMS25,$ECFMS27,$ECFMS28,$ECFMS29,
                $ECFMS35
```

See Also: $OPENENV for Open Modes, Access Codes, Formats
SPRM Ref: $DOPEN Vol.3 Sec. 3.3.1.1

Direct Random Access, Position

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DPOS     (work, fcb, fp, &err)
Result  : D$CCODE
Assign  : varRslt := $DPOS     (work, fcb, fp, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; $XCFMS01,$XCFMS07

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DPOS          Vol.3 Sec.  3.3.3.5
```

$DPOSEOF FUNCTION $DPOSEOF

Direct Random Access, Position to End of File

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$FAR

Syntax : **$DPOSEOF** ***(work, fcb)***
Result : *D$CCODE*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBD;**

If Error: **IF $DPOSEOF (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DPOSEOF Vol.3 Sec. 3.3.3.6

$DPOSNX FUNCTION $DPOSNX

Direct Sequential Access, Position to Next

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$FAR

Syntax : **$DPOSNX** ***(work, fcb, &err)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $DPOSNX** ***(work, fcb, &err);***
IF ***varRslt*** **&& D$ZFLAG**
THEN *("err" is exception code);*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBD;**
** Exit 3*: ***err*** BYTE; $XCFMS01

If Error: **IF** ***varRslt*** **&& D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DPOSNX Vol.3 Sec. 3.3.2.10

Direct Sequential Access, Position to Previous

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DPOSPV   (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $DPOSPV   (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
* Exit 3: err     BYTE; $XCFMS01

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $DPOSPV       Vol.3 Sec.  3.3.2.11
```

Direct Random Access, Read

```
Category: File Access Routine
Entered : 82 Jul 01               Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DREAD    (work, fcb, fp, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $DREAD (work, fcb, fp, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; $XCFMS01,$XCFMS02,$XCFMS07
* Exit 5: end   ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
SPRM Ref: $DREAD        Vol.3 Sec.  3.3.3.1
```

Direct Sequential Access, Read Current

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DREADCR  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $DREADCR (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
* Exit 3: err     BYTE; $XCFMS02 $XCFMS06
* Exit 4: end   ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
SPRM Ref: $DREADCR      Vol.3 Sec.  3.3.2.5
```

Direct Sequential Access, Read Next

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DREADNX  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $DREADNX  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
* Exit 3: err     BYTE; $XCFMS01 if EOF
* Exit 4: end   ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
SPRM Ref: $DREADNX      Vol.3 Sec.  3.3.2.3
```

Direct Sequential Access, Read Previous

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$FAR

Syntax  : $DREADPV  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $DREADPV  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
* Exit 3: err     BYTE; $XCFMS01 if Beginning of File
* Exit 4: end   ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
SPRM Ref: $DREADPV      Vol.3 Sec.  3.3.2.4
```

Direct Random Access, Rewrite

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$FAR

```
Syntax   : $DRWRT   (work, fcb, fp, &err, &end)
Result   : D$CCODE
Assign   : varRslt := $DRWRT (work, fcb, fp, &err, &end);
           IF varRslt && D$ZFLAG
           THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; $XCFMS01,$XCFMS02
* Exit 5: end   ^ CHAR; last byte read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS9,$ECFMS36
Remarks : Rewrites if record is already there.
          (no $DDEL required first)
SPRM Ref: $DRWRT        Vol.3 Sec.  3.3.3.4
```

Direct Sequential Access, Rewrite Current

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$FAR

Syntax : **$DRWRTCR** ***(work, fcb, &err, &end)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $DRWRTCR** ***(work, fcb, &err, &end);***
IF ***varRslt*** **&& D$ZFLAG**
THEN *("err" is exception code);*

Entry 1 :	***work***	**^ [256] BYTE;**	Paged buffer, D$BUFn
Entry 2 :	***fcb***	**^ $FCBD;**	
** Exit 3*:	***err***	**BYTE;**	$XCFMS02 $XCFMS06
** Exit 4*:	***end***	**^ CHAR;**	

If Error: **IF** ***varRslt*** **&& D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS9,$ECFMS36
SPRM Ref: $DRWRTCR Vol.3 Sec. 3.3.2.8

Direct Random Access, Write

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Jul 30
File    : D$FAR

Syntax  : $DWRITE  (work, fcb, fp, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $DWRITE (work,fcb,fp,&err,&end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBD;
Entry 3 : fp    ^ $FILEPTR;
* Exit 4: err     BYTE; $XCFMS05
* Exit 5: end   ^ CHAR; last byte read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS9,$ECFMS36
Remarks : Old Record on disk must have $LDEL characters
          (from $DDEL) if before $LEOF to prevent
          accidental overwrite.
SPRM Ref: $DWRITE       Vol.3 Sec.  3.3.3.2
```

$DWRITNX FUNCTION $DWRITNX

Direct Sequential Access, Write Next at EOF

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$FAR

Syntax : **$DWRITNX** ***(work, fcb, &end)***
Result : *D$CCODE*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBD;**
* *Exit 3*: ***end*** **^ CHAR;** last byte read from user area

If Error: IF **$DWRITNX (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10,$ECFMS3
Remarks : Always writes a $LEOF.
SPRM Ref: $DWRITNX Vol.3 Sec. 3.3.2.6

$DWRTEOF FUNCTION $DWRTEOF

Direct Sequential Access, Write End of File

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$FAR

Syntax : **$DWRTEOF** ***(work, fcb)***
Result : *D$CCODE*

Entry 1 : ***work*** **^ [256] BYTE;** Paged buffer, D$BUFn
Entry 2 : ***fcb*** **^ $FCBD;**

If Error: IF **$DWRTEOF (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $FMS
$ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS36
SPRM Ref: $DWRTEOF Vol.3 Sec. 3.3.2.9

ELSE FUNCTION ELSE

Part of IF THEN ELSE Execution Control

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **see IF function**
DASL Doc: 73 March 1982

ENTRY FUNCTION ENTRY

Declare Global Name; may be Referenced Externally

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

SYNTAX..: **ENTRY *name type [:= value];***

Remarks : May be used when defining variable (including function variables.
DASL Doc: 56,75,76,77 March 1982

Define Variable Type BYTE; Define Values 0 thru 8

Category: DASL Include Macro
Entered : 82 Jul 01 *Updated* : 84 Jul 01 |
File : D$INC

Syntax : **ENUM** ***(arg1, arg2,...arg9)***
Result : *BYTE*
Use : **TYPDEF *xvar* ENUM *(arg1, arg2,...arg9);*** or |
xvar* ENUM *(arg1, arg2,...arg9);

Remarks :
```
TYPDEF xvar ENUM (arg1, arg2,...arg9)
Equivalent to:
TYPDEF xvar BYTE;
DEFINE (arg1,0)
DEFINE (arg2,1)
...
DEFINE (arg9,8)
```

This defines "xvar" as a variable type BYTE and defines values 0 thru 8 to "arg1" thru "ar which may be assigned to the variable "xvar."

See Also: ENUMV for undefined types
DASL Doc: 34,90 March 1982

ENUMV FUNCTION ENUMV

Define Values Incrementing from Initial Value

Category: DASL Include Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$INC

Syntax : **ENUMV** ***(initValue, arg1, arg2,....arg8)***
Result : *none*

Remarks : Similar to ENUM except no type is defined.

```
ENUMV (100, arg1, arg2,....arg8)
Equivalent to:
DEFINE (arg1,100)
DEFINE (arg2,101)
...
DEFINE (arg8,107)
```

See Also: ENUM for variables with defined types
DASL Doc: 34,90 March 1982

$ENVDEL FUNCTION $ENVDEL

Delete Existing Environment

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVDEL** ***(name)***
Result : *D$CCODE*

Entry 1 : ***name*** **^ $ENVN;**

If Error: **IF $ENVDEL (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $UECNVD
$ERRC.$CODE = $UECENV1 (no such environment)
SPRM Ref: ENVDEL$ Vol.2 Sec. 2.2

Find Environment Data Match

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 19
File    : D$UFRENV

Syntax  : $ENVFNDM  (mask, start, &env, &prev)
Result  : D$CCODE

Entry 1 : mask    ^ CHAR;
Entry 2 : start ^ ^ $ENVT;
* Exit 3: env   ^ ^ $ENVT;
* Exit 4: prev  ^ ^ $ENVT;

If Error: IF  $ENVFNDM (...) && D$CFLAG
          THEN (no match found )
SPRM Ref: ENVFNDM$      Vol.2 Sec.  2.8
```

Insert New Environment

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRENV

Syntax : **$ENVINS** ***(env, after, &err)***
Result : *D$CCODE*
Use : **IF $ENVINS *(env, after, &err)* && D$CFLAG**
THEN *("err" is error code)* or **$ERMSG ( );**

Entry 1 : ***env*** **^ $ENVT;** New environment entry
Entry 2 : ***after*** **^ $ENVN;** Existing entry to precede new
** Exit 3*: ***err*** **BYTE;** If Error:
0 - invalid entry format
1 - no such existing entry "after"
2 - duplicate entry
3 - UET memory overflow

If Error: **IF $ENVINS (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $UECNVI
$ERRC.$CODE = $UECENV0,$UECENV1,$UECENV2,
$UECENV3
Remarks : "err" codes same as $ERRC.CODE
SPRM Ref: ENVINS$ Vol.2 Sec. 2.1

$ENVLGET FUNCTION $ENVLGET

Obtain Environment Entry Length

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVLGET** ***(env)***
Result : *UNSIGNED*

Entry 1 : ***env*** **^ $ENVT;** Name field of enviroment entry

If Error: **No Error Occurs**
SPRM Ref: ENVLGET$ Vol.2 Sec. 2.4

$ENVLOC FUNCTION $ENVLOC

Locate Existing Environment

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRENV

Syntax : **$ENVLOC** ***(name, &env, &prior)***
Result : *D$CCODE*

Entry 1 : ***name*** **^ $ENVN;** Name of entry to be located
** Exit 2*: ***env*** **^ ^ $ENVT;** Link field in located env
** Exit 3*: ***prior*** **^ ^ $ENVT;** Link field of prior env

If Error: **IF $ENVLOC (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $UECNVL
$ERRC.$CODE = $UECENV1 (item not found)
SPRM Ref: ENVLOC$ Vol.2 Sec. 2.3

Position to Environment Data Item in an Env Entry

Category:	User Function Routine
Entered :	82 Jul 01 *Updated* : 83 Jul 10
File :	D$UFRENV
Syntax :	**$ENVPDAT** ***(env)***
Result :	^ *CHAR*
Entry 1 :	***env*** ^ ^ **$ENVT;** Env entry link field ($ENVLOC)
If Error:	**No Error Occurs**
Remarks :	Equivalent to $ENVPNET
See Also:	$ENVPNET for positioning to net name
SPRM Ref:	ENVPDAT$ Vol.2 Sec. 2.5

Prepare Open of Environment's Catalog File

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$UFRENV

Syntax : **$ENVPEEL *(env, openpt, &end)***
Result : *D$CCODE*

Entry 1 : ***env*** **^ ^ $ENVT;** Env entry link field $ENVLOC
Entry 2 : ***openpt*** **^ $OPENPT;** Specifies user area + PFDB
** Exit 3*: ***end*** **^ CHAR;** Last byte stored in user area

If Error: **IF $ENVPEEL (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = $UECNVP
$ERRC.$CODE = $UECENVO
Remarks : Useful prior to calling $OPENENV and then calls to $FILES for accessing files in the environment by file name.
Prepares $OPENPT specified for $OPENENV:
1. Copies all "env" data but lowest level HSI to $OPENPT.$OTENV ^
2. Copies lowest level HSI name in "env" to $OPENPT.$OTFILE ^ with spaces in extension.
SPRM Ref: ENVPEEL$ Vol.2 Sec. 2.7

$ENVPHSI FUNCTION $ENVPHSI

Position to HSI Name in an Environment Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPHSI** ***(env)***
Result : ^ *CHAR*

Entry 1 : ***env*** **^ ^ $ENVT;**

If Error: **No Error Occurs**
SPRM Ref: ENVPHSI$ Vol.2 Sec. 2.5

$ENVPLOP FUNCTION $ENVPLOP

Env, Position to UET Link In Open Parameter Table

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPLOP** ***(openpt)***
Result : ^ ^ *$ENVT*

Entry 1 : ***openpt*** **^ $OPENPT;**

If Error: **No Error Occurs**
SPRM Ref: ENVPLOP$ Vol.2 Sec. 2.6

$ENVPNAM FUNCTION $ENVPNAM

Position to Environment Name in an Env Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPNAM** ***(env)***
Result : *^ $ENVT*

Entry 1 : ***env*** **^ ^ $ENVT;**

If Error: **No Error Occurs**
SPRM Ref: ENVPNAM$ Vol.2 Sec. 2.5

$ENVPNET FUNCTION $ENVPNET

Position to Net Name in an Environment Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRENV

Syntax : **$ENVPNET** ***(env)***
Result : *^ CHAR*

Entry 1 : ***env*** **^ ^ $ENVT;**

If Error: **No Error Occurs**
See Also: $ENVPDAT for positioning to data item
SPRM Ref: ENVPNET$ Vol.2 Sec. 2.5

$ENVPNOD FUNCTION $ENVPNOD

Position to Node Name in an Environment Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPNOD** ***(env)***
Result : *^ CHAR*

Entry 1 : ***env*** **^ ^ $ENVT;**

If Error: **No Error Occurs**
SPRM Ref: ENVPNOD$ Vol.2 Sec. 2.5

$ENVPPAS FUNCTION $ENVPPAS

Position to First Password in Environment Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPPAS** ***(env)***
Result : *^ $PACKPW*

Entry 1 : ***env*** **^ ^ $ENVT;**

If Error: **No Error Occurs**
SPRM Ref: ENVPPAS$ Vol.2 Sec. 2.5

$ENVPRES FUNCTION $ENVPRES

Position to Resource Name in Environment Entry

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRENV

Syntax : **$ENVPRES** *(env)*
Result : *^ CHAR*

Entry 1 : ***env*** **^ ^** $ENVT;

If Error: **No Error Occurs**
SPRM Ref: ENVPRES$ Vol.2 Sec. 2.5

$ERMSG FUNCTION $ERMSG

Display RMS Minimum Error Message

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMS

Syntax : **$ERMSG** *()*
Result : *Does not RETURN (goes to $ERROR)*

If Error: **No Error Occurs**
Remarks : The $ERRC 2 byte variable contains the Function and Error codes obtained from the previous function that was called and detected an error.
SPRM Ref: ERMSG$ Vol.2 Sec. 3.3.4

$ERROR FUNCTION $ERROR

Abort a Program

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$RMS

Syntax : **$ERROR** *()*
Result : *Does not RETURN*

If Error: **No Error Occurs**
SPRM Ref: $ERROR Vol.4 Sec. 4.4

$EXIT FUNCTION $EXIT

Exit a Program

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 19
File : D$RMS

Syntax : **$EXIT** *()*
Result : *Does not RETURN*

If Error: **No Error Occurs**
SPRM Ref: $EXIT Vol.4 Sec. 4.3

EXTERN FUNCTION EXTERN

Declare a Name Defined in Another Program Module

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **EXTERN** ***name type ;***
DASL Doc: 56,75,76,77 March 1982

FAST FUNCTION FAST

Future Code Generators; Var. Resides in Register

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 83 Jul 30

Syntax : FAST ***varName type ;***
Remarks : Also used by TRAP functions to declare
TRAP routine code as FAST CODE.

$FDPACK FUNCTION $FDPACK

Numeric, Pack Two Decimal Numbers

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRNUM

Syntax : **$FDPACK** ***(first, second, &packed)***
Result : *D$CCODE*

Entry 1 :	***first***	**CHAR;**
Entry 2 :	***second***	**CHAR;**
** Exit 3*:	***packed***	**CHAR;**

If Error: **IF $FDPACK (...) && D$CFLAG**
THEN *(not packable);*
SPRM Ref: FDPACK$ Vol.2 Sec. 4.18

Numeric, Unpack Character Into Two ASCII Digits

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRNUM

Syntax : **$FDUNPAK** ***(packed, &first, &second)***
Result : *none*

Entry 1 :	***packed***	**CHAR;**
** Exit 2*:	***first***	**CHAR;**
** Exit 3*:	***second***	**CHAR;**

If Error: **No Error Occurs**
SPRM Ref: FDUNPAK$ Vol.2 Sec. 4.19

Convert File Format Codes to 4-Byte ASCII String

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRGEN

Syntax : $FILEFMT ***(format, &string)***
Result : *BYTE*

Entry 1 : **format** BYTE; File format code ($FFMT...)
* *Exit 2*: **string** ^ CHAR; 4-Byte ASCII string

If Error: **No Error Occurs**
SPRM Ref: FILEFMT$ Vol.2 Sec. 8.4

Environment, Obtain the Next File Name in Catalog

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRENV...also D$RMSIO for $FILEINFO TYPDEF

Syntax : **$$FILENAM *(&fileInfo)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $$FILENAM *(&fileInfo)*;**
IF *varRslt* && D$CFLAG=0 & *varRslt* && D$ZFLAG=0
THEN *("fileInfo" entry obtained);*
IF *varRslt* && D$CFLAG=0 & *varRslt* && D$ZFLAG
THEN *(no more files in catalog,catalog closed);*

* *Exit 1*: ***fileInfo*** **^ $FILEINFO;** From $FILES($FILENAM)

If Error: **IF *varRslt* && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$FILES error in $FILES
$ERRC.$FUNC = $UECFIL error in calling seq.
$ERRC.$CODE = $UECFIL1,$UECFIL2
Remarks : Call $$FILEPCN first to begin access to a new catalog. Then call $$FILENAM successively until the catalog is closed. Call $$FILEPCN to access another catalog.
See Also: $$FILEPCN to see how to access catalog.
SPRM Ref: FILENAM$ Vol.2 Sec. 2.14.3

Environment, Open Catalog File and Obtain PCNs

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Jul 10
File    : D$UFRENV

Syntax  : $$FILEPCN (env, start, end, &num)
Result  : D$CCODE
Assign  : varRslt := $$FILEPCN (env, start, end, &num);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN ("num" PCN's were obtained);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (no PCN's obtained; catalog empty);

Entry 1 : env    ^ ^ $ENVT; UET entry link field
Entry 2 : start  ^ UNSIGNED; Start of PCN storage area
                             2 bytes per PCN required
Entry 3 : end    ^ UNSIGNED; End of PCN storage area
* Exit 4: num      UNSIGNED; Number of PCN's obtained

If Error: IF varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = SC$OPENENV  failure in $OPENENV
          $ERRC.$FUNC = SC$FILES  $FILES($FILEPCN) error
          $ERRC.$FUNC = $UECFIL
          $ERRC.$CODE = $UECFILO
Remarks : Performs the following actions:
          1) $ENVPEEL creates surrogate env to open cat
          2) $OPENENV on catalog
          3) $FILES in $FILEPCN mode to obtain all PCN's
          4) Sorts PCN's for optimum disk head position
          5) Sets initial values for calls to $$FILENAM

          NOTE:
          PCN's delivered are based on hashed value of
          the HSI level, thus non-members of the catalog
          may be delivered. Zero PCN's delivered does
          guarantee the catalog is empty.

See Also: $$FILEPCU,$$FILENAM for obtaining next entry
          in catalog and alternative entry method.
SPRM Ref: FILEPCN$      Vol.2 Sec.  2.14.1
```

Environment, Special Entry to $$FILEPCN

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Jul 10
File    : D$UFRENV

Syntax  : $$FILEPCU (pfdb, start, end, &num)
Result  : D$CCODE
Assign  : varRslt := $$FILEPCU (pfdb, start, end, &num);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN ("num" PCN's were obtained);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (no PCN's obtained; catalog empty);

Entry 1 : pfdb   ^ $PFDB; Opened catalog file PFDB
Entry 2 : start  ^ UNSIGNED; Start of PCN storage area
Entry 3 : end    ^ UNSIGNED; End of PCN storage area
* Exit 4: num      UNSIGNED; Number of PCN's obtained

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$FILES $FILES($FILEPCN) error
          $ERRC.$FUNC = $UECFIL
          $ERRC.$CODE = $UECFILO
Remarks : May be used instead of $$FILEPCN when
          Catalog File is already opened.
See Also: $$FILEPCN for opening Catalog File.
SPRM Ref: FILEPCU$       Vol.2 Sec.  2.14.2
```

Multi-Resource, Obtain Disk File Information

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $FILES    (mode, filesTbl)
Result  : D$CCODE

Entry 1 : mode      BYTE; $FILEPCN,$FILENAM,$FILECHK
Entry 2 : filesTbl  ^ $FILESTBL; Redefined PFDB

If Error: IF  $FILES (...) && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = SC$FILES
          $ERRC.$CODE = Single- and multi-file            |*
                $ECUMAV,$ECSI001,$ECSI005,$ECSI010,       |*
                $ECSI021,$ECSI034,$ECSI045                |*

                        Multi-file only                   |*
                $ECSI004,$ECSI007,$ECSI012,$ECSI014,      |*
                $ECSI036,$ECSI038,$ECSI060                |*
See Also: $GETSFI uses $FILES. See  $ENVPEEL for
          preparing environment prior to usage.
SPRM Ref: $FILES        Vol.4 Sec.  8.4.4
SPRM Ref: $FILESTBL     Vol.4 Sec.  2.2.3 SPRM
SPRM Ref:  FILES$       Vol.2 Sec.  2.14 SPRM
```

Multi-Resource, Format a Unit on the Disk

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25  |*
File    : D$RMSIO

Syntax  : $FORMAT   (pfdb)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;

If Error: IF  $FORMAT (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$FORMAT
          $ERRC.$CODE = All resources                    |*
                $ECUMAV,$ECSI001,$ECSI002,$ECSI003,      |*
                $ECSI004,$ECSI005,$ECSI007,$ECSI010,     |*
                $ECSI017,$ECSI039                        |*

                        Disk resources                   |*
                $ECSI006,$ECSI009,$ECSI015,$ECSI016,     |*
                $ECSI018,$ECSI045,$ECSI046,$ECSI058      |*

                        Tape resources                   |*
                $ECSI006,$ECSI009,$ECSI016,$ECSI018,     |*
                $ECSI046                                 |*
SPRM Ref: $FORMAT       Vol.4 Sec.  8.5.8
```

$FSCAN FUNCTION $FSCAN

CmdInt, Compress a $FILESPK

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRSCAN

Syntax : $FSCAN *(sfent, output)*
Result : *BYTE*

Entry 1 : ***sfent*** ^ $NAMET;
Entry 2 : ***output*** ^ CHAR;

If Error: **No Error Occurs**

Remarks : extracts name, extension, and environmment from FILE SPECIFICATION TABLE, compresses into single string, ready to insert into messages or display headings.
SPRM Ref: FSCAN$ Vol.2 Sec. 6.1.4

$GDATTIM FUNCTION $GDATTIM

Obtain Current ASCII Date/Time String

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRGEN

Syntax : **$GDATTIM** ***(months, &string)***
Result : *BYTE*

Entry 1 : ***months*** ^ [12] [3] CHAR; Month-Name Table
** Exit 2*: ***string*** ^ CHAR; DD MMM YYYY HH:MM

If Error: **No Error Occurs**

See Also: $DATETIM. Both Use CVSTIME$ as subroutine
SPRM Ref: GDATTIM$ Vol.2 Sec. 8.8

$GENSMSK FUNCTION $GENSMSK

CmdInt, Generate Generic Scanning Masks

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Apr 23
File    : D$UFRSCAN

Syntax  : $GENSMSK  (input, output)
Result  : D$CCODE

Entry 1 : input    ^ $NAMEEXTENV;
Entry 2 : output   ^ $NAMEEXTENV;

If Error: IF  $GENSMSK (...) && D$CFLAG
          THEN (invalid masks);
Remarks : generates internal tables for name scanning
SPRM Ref: GENSMSK$      Vol.2 Sec.  6.6.1
```

$GENSTST FUNCTION $GENSTST

Command Int., Name Test-Under-Mask and Generate

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Apr 23
File    : D$UFRSCAN

Syntax  : $GENSTST  (file, &output)
Result  : D$CCODE
Assign  : varRslt := $GENSTST  (file, &output);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN (no match; test failed);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (match; test succesful);

Entry 1 : file     ^ $NAMEEXTENV;
* Exit 2: output   ^ $NAMEEXTENV;

If Error: IF varRslt && D$CFLAG
          THEN (illegal output name generated);
Remarks : tests a single complete file name. If previous
          call to $GENSMSK specified output mask, then
          will generate output name...further calls to
          $GENSMSK will generate new masks to test
SPRM Ref: GENSTST$      Vol 2 Sec.  6.6.2
```

Workstation-IF, Get Response from CHAIN File / WS

```
Category: User Function Routine
Entered : 82 Jul 01                  Updated : 84 Jul 01 |*
File    : D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)

Syntax  : $GETCHN   (line, mode, &length, &hor, &ver)
Result  : D$CCODE

Entry 1 : line    ^ CHAR; Pointer to input area
Entry 2 : mode      BYTE; $WSIOMODE: $WSM....
Entry 3 : length    BYTE; Input area (1 for $WSENTK)
Entry 4 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 5 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 3: length    BYTE; Initial length - keys input
* Exit 4: hor       BYTE; cursor position of $WSENTK
* Exit 5: ver       BYTE; cursor position of $WSENTK


If Error: IF $GETCHN (...) && D$CFLAG THEN $ERMSG( );  |*
          $WSIO errors result in call to $ERMSG.       |*
See Also: $GETLINE for other workstation Get Response
          $WCONFIG for screen size.
SPRM Ref: GETCHN$        Vol.2 Sec.  7.4
```

Workstation-IF, Timeout for GETCSTK$ and GETCHN$

Category:	User Function Routine
Entered :	82 Jul 01 *Updated* : 83 Jul 01
File :	D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)
Syntax :	$GETCHTO ***(line, mode, &length, &hor, &ver)***
Result :	*D$CCODE*

Entry 1 :	***line***	^	**CHAR;**	Pointer to input area
Entry 2 :	***mode***		**BYTE;**	$WSIOMODE: $WSM....
Entry 3 :	***length***		**BYTE;**	Input area (1 for $WSENTK)
Entry 4 :	***hor***		**BYTE;**	Initial hor curs pos($WSLC)
Entry 5 :	***ver***		**BYTE;**	Initial ver curs pos($WSBL-x)
* *Exit 3*:	***length***		**BYTE;**	Initial length - keys input
* *Exit 4*:	***hor***		**BYTE;**	cursor position of $WSENTK
* *Exit 5*:	***ver***		**BYTE;**	cursor position of $WSENTK

If Error: **IF $GETCHTO (...) && D$CFLAG THEN $ERMSG( );**
$WSIO errors result in call to $ERMSG.

See Also: $GETLINE for error messages
$WCONFIG for screen size.

SPRM Ref: GETCHTO$ Vol.2 Sec. 7.7

Workstation-IF, Get Response from Command Stack / WS

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)

Syntax  : $GETCSTK  (line, mode, &length, &hor, &ver)
Result  : D$CCODE

Entry 1 : line    ^ CHAR; Pointer to input area
Entry 2 : mode      BYTE; $WSIOMODE: $WSM....
Entry 3 : length    BYTE; Input area (1 for $WSENTK)
Entry 4 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 5 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 3: length    BYTE; Initial length - keys input
* Exit 4: hor       BYTE; cursor position of $WSENTK
* Exit 5: ver       BYTE; cursor position of $WSENTK

If Error: IF $GETCSTK (...) && D$CFLAG THEN $ERMSG( );
          $WSIO errors result in call to $ERMSG.
See Also: $GETLINE for error messages.
          $WCONFIG for screen size.
SPRM Ref: GETCSTK$      Vol.2 Sec.  7.4
```

Workstation-IF, Timeout for GETCSTK$ and GETCHN$

Category:	User Function Routine
Entered :	82 Jul 01 *Updated* : 83 Jul 01
File :	D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)

Syntax :	$GETCSTO ***(line, mode, &length, &hor, &ver)***
Result :	*D$CCODE*

Entry 1 :	***line***	^ **CHAR**;	Pointer to input area
Entry 2 :	***mode***	**BYTE**;	$WSIOMODE: $WSM....
Entry 3 :	***length***	**BYTE**;	Input area (1 for $WSENTK)
Entry 4 :	***hor***	**BYTE**;	Initial hor curs pos($WSLC)
Entry 5 :	***ver***	**BYTE**;	Initial ver curs pos($WSBL-x)
** Exit 3*:	***length***	**BYTE**;	(Initial length)-(keys input)
** Exit 4*:	***hor***	**BYTE**;	cursor position of $WSENTK
** Exit 5*:	***ver***	**BYTE**;	cursor position of $WSENTK

If Error:	**IF $GETCSTO (...) && D$CFLAG THEN $ERMSG( );**
	$WSIO errors result in call to $ERMSG.
See Also:	$GETLINE for error messages.
	$WCONFIG for screen size.
SPRM Ref:	GETCSTO$ Vol.2 Sec. 7.7

Obtain Current System Time

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Aug 08   |*
File    : D$RMSGEN

Syntax  : $GETIME   (dest, utc)
Result  : D$CCODE

Entry 1 : dest   ^ $SYSTIME; 6 bytes,(seconds since
                              beginning 1901, & 8ms.ctr)
Entry 2 : utc      BOOLEAN; Use Alternate SC if TRUE

If Error: IF  $GETIME (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$GETIME
          $ERRC.$CODE = $ECUMAV
Remarks : When you use the alternate System Call, you   |*
          must use the following syntax:                |*
          $GETIME (<^$SYSTIME> dest, TRUE)              |*
          where:                                        |*
            dest is a ^$SYSTINFO                        |*
SPRM Ref: $GETIME      Vol.4 Sec.  3.1
```

Workstation-IF, Get Response from Stack, CHAIN, or WS

Category:	User Function Routine
Entered :	82 Jul 01 *Updated* : 83 Jul 10
File :	D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)
Syntax :	$GETLINE ***(line, mode, &length, &hor, &ver)***
Result :	*D$CCODE*

Entry 1 :	***line***	**^ CHAR**;	Pointer to input area
Entry 2 :	***mode***	BYTE;	$WSIOMODE: $WSM....
Entry 3 :	***length***	BYTE;	Input area (1 for $WSENTK)
Entry 4 :	***hor***	BYTE;	Initial hor curs pos($WSLC)
Entry 5 :	***ver***	BYTE;	Initial ver curs pos($WSBL-x)
** Exit 3*:	***length***	BYTE;	Initial length - keys input
** Exit 4*:	***hor***	BYTE;	cursor position of $WSENTK
** Exit 5*:	***ver***	BYTE;	cursor position of $WSENTK

If Error: **IF $GETLINE (...) && D$CFLAG THEN $ERMSG( );**
$ERRC.$FUNC = $UECCHN
$ERRC.$CODE = $UECCHN0,$UECCHN1
$ERRC.$FUNC = $UECGLN
$ERRC.$CODE = $UECGLN0,$UECGLN1,$UECGLN2
$ERRC.$FUNC = SC$OPENENV
THEN (error during LOG or CHAIN file open)
$ERRC.$FUNC = SC$SECR or SC$SECW
THEN (error during CHAIN or LOG I/O)

$WSIO errors result in call to $ERMSG

See Also: $GETCHN,$GETCHTO,$GETCSTK,$GETCSTO,$GETLINE, $GETLNTO,$KEYIN,$KEYINTO for other WorkStation response gathering. $WCONFIG for screen size.

SPRM Ref: GETLINE$ Vol.2 Sec. 7.2

Workstation-IF, Timeout Controlled Version of $GETLINE

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF)

Syntax : **$GETLNTO** ***(line, mode, &length, &hor, &ver)***
Result : *D$CCODE*

Entry 1 :	***line***	**^ CHAR;**	Pointer to input area
Entry 2 :	***mode***	**BYTE;**	$WSIOMODE: $WSM....
Entry 3 :	***length***	**BYTE;**	Input area (1 for $WSENTK)
Entry 4 :	***hor***	**BYTE;**	Initial hor curs pos($WS.L)
Entry 5 :	***ver***	**BYTE;**	Initial ver curs pos($WSBL-x)
* *Exit 3*:	***length***	**BYTE;**	(Initial length)-(keys input)
* *Exit 4*:	***hor***	**BYTE;**	cursor position of $WSENTK
* *Exit 5*:	***ver***	**BYTE;**	cursor position of $WSENTK

If Error: **IF $GETLNTO (...) && D$CFLAG THEN $ERMSG( );**
$WSIO errors result in call to $ERMSG.
See Also: $GETLINE for error messages
$WCONFIG for screen size.
SPRM Ref: GETLNTO$ Vol.2 Sec. 7.5

Environment; Obtain, Compress Password from Keyin

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRENV

Syntax  : $GETPASS  (output, hor, ver, &invalid)
Result  : D$CCODE
Assign  : varRslt := $GETPASS (output,hor,ver,&invalid);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN (valid password entered);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (null password entered);

Entry 1 : output    ^ $PACKPW;
Entry 2 : hor         BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver         BYTE; Init ver curs pos($WSBL-x)
* Exit 4: invalid   ^ CHAR;

If Error: IF varRslt && D$CFLAG THEN (invalid
          password entered);
See Also: $WCONFIG for screen size.
SPRM Ref: GETPASS$      Vol.2 Sec.  2.9
```

Obtain Symbolic File Identification

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Aug 08
File : D$RMSIO

Syntax : **$GETSFI** ***(filesTbl)***
Result : *D$CCODE*

Entry 1 : ***filesTbl*** **^ $FILESTBL**; Redefined $PFDB

If Error: **IF $GETSFI (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$FILES
$ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,
$ECSI010
Remarks : useful for obtaining symbolic names & HSIs, |*
after an error is detected, for use in |*
message display |*
See Also: Calls $FILES with mode $FILEGET
$PCRFIEI and $SFITABLE in TYPDEF for table built

SPRM Ref:	$GETSFI	Vol.4 Sec. 7.4.2
SPRM Ref:	$GETSFI	Vol.4 Sec. 8.4.3
SPRM Ref:	$GETSFI	Vol.4 Sec. 9.1.1.4
SPRM Ref:	$GETSFI	Vol.4 Sec. 9.2.1.2
SPRM Ref:	$GETSFI	Vol.4 Sec. 9.3.1.2
SPRM Ref:	$GETSFI	Vol.4 Sec. 9.4.1.2
SPRM Ref:	$GETSFI	Vol.4 Sec. 10.1.2

$GLUTEN FUNCTION $GLUTEN

Get Last User Task Error Number

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$RMSPROG

Syntax : **$GLUTEN** ***(&luten, &tranPsk, &tranTb)***
Result : *D$CCODE*

* *Exit 1*: ***luten*** **BYTE;** Error codes $UTE....?
* *Exit 2*: ***tranPsk*** **BYTE;**
* *Exit 3*: ***tranTb*** **^ [128] CHAR;**

If Error: **No Error Occurs**
SPRM Ref: $GLUTEN Vol.4 Sec. 4.5.8

GOTO FUNCTION GOTO

Transfer Control to Labled Statement in Same Func

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **GOTO** ***identifier;...identifier : statement;***
DASL Doc: 74,75 March 1982

ISAM, Close

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$FAR

Syntax  : $ICLOSE   (work, fcb, mode)
Result  : D$CCODE

Entry 1 : work ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb  ^ $FCBIS;
Entry 3 : mode   BYTE; close mode ($CM...)

If Error: IF  $ICLOSE (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS4,
                        $ECFMS5,$ECFMS6
SPRM Ref: $ICLOSE       Vol.3 Sec.  4.3.1.3
```

ISAM Random, Delete

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IDEL  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IDEL  (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBIS;
* Exit 3: err      BYTE; $XCFMS01,$XCFMS02,$XCFMS03

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS14,
                        $ECFMS36,$ECFMS38
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022,$ECLI024
SPRM Ref: $IDEL         Vol.3 Sec.  4.3.3.3
```

ISAM Sequential, Delete Current Record Key, Data

```
Category: File Access Routine
Entered : 82 Jul 01              Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IDELCR  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IDELCR (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBIS;
* Exit 3: err     BYTE; $XCFMS01,$XCFMS02,$XCFMS07

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS36,
                        $ECFMS38
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IDELCR       Vol.3 Sec.  4.3.2.4
```

ISAM Random, Delete Record's Key

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IDELK  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IDELK (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBIS;
* Exit 3: err      BYTE; $XCFMS03

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS14,
                        $ECFMS38
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022,$ECLI024
SPRM Ref: $IDELK        Vol.3 Sec.  4.3.3.4
```

IF — FUNCTION — IF

Execute THEN Statement if Expression Non-zero, etc

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : IF ***expression*** THEN ***statement***

```
          [ ELSE statment ];
          also:
          IF expression THEN {
              statement;
              statement;
              }              (No semicolon)
            ELSE {
              statement;
              statement;
              };             (Semicolon)
```

DASL Doc: 73 March 1982

IFELSE — FUNCTION — IFELSE

If First 2 Strings are Equal Result is 3rd,Else 4

Category: DASL Compiler Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 10

Syntax : IFELSE ***(strg1,strg2,equalStrg,unequalStrg)***
Result : *string*

See Also: DEFINE for evaluation suppression marks and parameter definitions
DASL Doc: 21,29,33,80 March 1982

ISAM Random, Insert

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IINS  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IINS (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work  ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb   ^ $FCBIS;
* Exit 3: err     BYTE; $XCFMS04

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS14,
                        $ECFMS38
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI020,$ECLI022,$ECLI024
SPRM Ref: $IINS         Vol.3 Sec.  4.3.3.5
```

ISAM, I/O Clear

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$FAR

Syntax  : $IIOCLR   (work, fcb)
Result  : D$CCODE

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBIS;

If Error: IF  $IIOCLR (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
SPRM Ref: $IIOCLR       Vol.3 Sec.  4.3.1.4
```

INCLUDE FUNCTION INCLUDE

Obtain Program Input Lines from Specified File

Category: DASL Compiler Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : **INCLUDE** ***(fileSpec)***
Remarks : Similar to the use of INCLUDE in SNAP but the syntax is different;
In DASL parentheses are required to mark the argument of the macro.

The order in which files are included must maintain upward visibility; variables and values defined in other include files must be included BEFORE files in which they are referenced.

DASL Doc: 1,2,8-10,29,82 March 1982

INCR FUNCTION INCR

Produce a Value by Incrementing the Argument by 1

Category: DASL Compiler Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 10

Syntax : **INCR** ***(number)***
Result : *decimal number string*

See Also: DEFINE for evaluation suppression marks and parameter definitions
DASL Doc: 29,31,32,80-81 March 1982

Obtain System Configuration Information

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25  |*
File    : D$RMSGEN

Syntax  : $INFO (mode, dest, max, netnode, &end, &rem)
Result  : D$CCODE

Entry 1 : mode        BYTE; $INFO modes

Entry 2 : dest     ^ $INFOITEM; Output area
Entry 3 : max        BYTE; Maximum items to return
Entry 4 : netnode  ^ CHAR; Net & Node name string
* Exit 5: end      ^ $INFOITEM; Past Last Item            |*
                              remaining                   |*
* Exit 6: rem        BYTE; Number of items remaining      |*

If Error: IF  $INFO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$INFO
          $ERRC.$CODE = $ECUMAV,$ECSI016,$ECSI021,        |*
                $ECSI028,$ECSI034,$ECSI035,$ECSI036,      |*
                $ECSI045,$ECSI055                         |*
Remarks : Used to obtain  information and symbolic
          names associated with: Node, Resources,
          Resource Connection Link, Task, Controller
          Variables, Resource Utilization Data,
          System Startup Time,
          Name Delimiter Characters.
          $INFO Modes:      015         012        013
          014$ILINKND, $ILINKAL, $INODEND, $INODEAL,
          017 $IMYNODE, $IRSFND,06 $IRSFAL,07 $IRMFND,04
           05 $IRMFAL,  $ITASKND, $ITASKAL, $ITASKME,016
           03 $ISTARTT        010        011
SPRM Ref: $INFO         Vol.4 Sec.  3.4
```

ISAM, Open

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$FAR

Syntax  : $IOPEN  (work, fcb, mode, openpt, name)
Result  : D$CCODE

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBIS;
Entry 3 : mode     BYTE; $OM...
Entry 4 : openpt ^ $OPENPT;
Entry 5 : name   ^ $NAMEEXTENV;

If Error: IF  $IOPEN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS1,$ECFMS2,$ECFMS3,
               $ECFMS4,$ECFMS5,$ECFMS6,$ECFMS7,
               $ECFMS11,$ECFMS12,$ECFMS13,$ECFMS19,
               $ECFMS20,$ECFMS21,$ECFMS27
See Also: $OPENENV for Open Modes, Access Codes,
          Formats
SPRM Ref: $IOPEN        Vol.3 Sec.  4.3.1.1
```

ISAM Random, Position

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IPOS  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IPOS  (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb  ^ $FCBIS;
* Exit 3: err    BYTE; $XCFMS01,$XCFMS03,$XCFMS07

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
          $ERRC.$FUNC = $LRIASM
          $ERRC.$CODE = $ECLI022,$ECLI024
SPRM Ref: $IPOS          Vol.3 Sec.  4.3.3.7
```

ISAM Sequential, Position to Key Previous

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IPOSKP  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IPOSKP (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb  ^ $FCBIS;
* Exit 3: err    BYTE; $XCFMS01 $XCFMS07

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IPOSKP       Vol.3 Sec.  4.3.2.7
```

ISAM Position to Next Key Sequential Record

```
Category: File Access Routine
Entered : 82 Jul 01              Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IPOSKS  (work, fcb, &err)
Result  : D$CCODE
Assign  : varRslt := $IPOSKS  (work, fcb, &err);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb  ^ $FCBIS;
* Exit 3: err    BYTE; $XCFMS01,$XCFMS02,$XCFMS07

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IPOSKS       Vol.3 Sec.  4.3.2.6
```

ISAM, Prepare File

Category: File Access Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$FAR

Syntax : **$IPREP** ***(work, fcb, openpt, name)***
Result : *D$CCODE*

Entry 1 :	***work***	**^**	**[256] BYTE;** Paged buffer, D$BUFn
Entry 2 :	***fcb***	**^**	**$FCBIS;**
Entry 3 :	***openpt***	**^**	**$OPENPT;**
Entry 4 :	***name***	**^**	**$NAMEEXTENV;**

```
If Error: IF  $IPREP (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS1,$ECFMS2,$ECFMS3,
                $ECFMS4,$ECFMS5,$ECFMS6,$ECFMS7,
                $ECFMS11,$ECFMS12,$ECFMS13,$ECFMS19,
                $ECFMS20,$ECFMS21,$ECFMS39,$ECFMS40
SPRM Ref: $IPREP        Vol.3 Sec.  4.3.1.2
```

ISAM Random, Read

```
Category: File Access Routine
Entered : 82 Jul 01                  Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IREAD  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IREAD (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit.3: err       BYTE; $XCFMS01,$XFCMS02,$XCFMS03
* Exit.4: end     ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI024
SPRM Ref: $IREAD        Vol.3 Sec.  4.3.3.1
```

ISAM Sequential, Read Current

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IREADCR  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IREADCR  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work   ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb    ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS01,$XCFMS02,$XCFMS07
* Exit 4: end    ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IREADCR      Vol.3 Sec.  4.3.2.3
```

ISAM Sequential, Read Key Previous

```
Category: File Access Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IREADKP  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IREADKP  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS01,$XCFMS02,$XCFMS07
* Exit 4: end     ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IREADKP      Vol.3 Sec.  4.3.2.2
```

ISAM Sequential, Read Key Sequential

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IREADKS  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IREADKS  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS01,$XCFMS02,$XCFMS07
* Exit 4: end     ^ CHAR; last byte stored in user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS10
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
          $ERRC.$FUNC = SC$SECR, SC$SECWAIT
          $ERRC.$CODE = (see $SECR, $SECWAIT)
SPRM Ref: $IREADKS      Vol.3 Sec.  4.3.2.1
```

ISAM Random, Rewrite

```
Category: File Access Routine
Entered : 82 Jul 01                    Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IRWRT  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IRWRT  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS01,$XCFMS02,$XCFMS03
* Exit 4: end     ^ CHAR; last byte read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS9,
                        $ECFMS36
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022,$ECLI024
SPRM Ref: $IRWRT        Vol.3 Sec.  4.3.3.6
```

ISAM Sequential, Rewrite Current

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IRWRTCR  (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IRWRTCR  (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS01,$XCFMS02,$XCFMS07
* Exit 4: end     ^ CHAR; last byte read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS2,$ECFMS9,$ECFMS36
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI022
SPRM Ref: $IRWRTCR      Vol.3 Sec.  4.3.2.5
```

ISAM Random, Write

```
Category: File Access Routine
Entered : 82 Jul 01                Updated : 83 Apr 02
File    : D$FAR

Syntax  : $IWRITE   (work, fcb, &err, &end)
Result  : D$CCODE
Assign  : varRslt := $IWRITE   (work, fcb, &err, &end);
          IF varRslt && D$ZFLAG
          THEN ( "err" is exception code );

Entry 1 : work    ^ [256] BYTE; Paged  buffer, D$BUFn
Entry 2 : fcb     ^ $FCBIS;
* Exit 3: err       BYTE; $XCFMS04
* Exit 4: end     ^ CHAR; Last char read from user area

If Error: IF  varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = $FMS
          $ERRC.$CODE = $ECFMS0,$ECFMS9,$ECFMS14,$ECFMS36
                        $ECFMS38
          $ERRC.$FUNC = $LRISAM
          $ERRC.$CODE = $ECLI020,$ECLI022,$ECLI024
SPRM Ref: $IWRITE       Vol.3 Sec.  4.3.3.2
```

Workstation-IF, Obtain One Translated Character

```
Category: User Function Routine
Entered : 82 Jul 01               Updated : 84 Jul 01
File    : D$UFRWS

Syntax  : $KEYCHAR (hor, ver, &char)
Result  : D$CCODE
Assign  : varRslt := $KEYCHAR (hor, ver, &char);
          IF varRslt && D$CFLAG=0 & varRslt && D$SFLAG=0
          THEN ("char" is translated character);
          IF varRslt && D$CFLAG=0 & varRslt && D$SFLAG
          THEN ("char" is untranslated control code);

Entry 1 : hor     BYTE; $WSLC to $WSRC+
Entry 2 : ver     BYTE; x to $WSBL
* Exit 3: char    CHAR;

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
Remarks : System call overhead very high for single
          character...should be used only when
          absolutely necessary.
See Also: $WSIO for error messages
          $WCONFIG for screen size.
          LISTS section: $WS... Workstation Keyboard   |*
          Codes                                        |*
SPRM Ref: KEYCHAR$      Vol.2 Sec.  7.8
```

$KEYCLR FUNCTION $KEYCLR

Workstation-IF, Clear the Keyin FIFO

```
Category: User Function Routine
Entered : 82 Jul 01                    Updated : 82 Dec 01
File    : D$UFRWS

Syntax  : $KEYCLR   ()
Result  : none

If Error: No Error Occurs
SPRM Ref: KEYCLR$      Vol.2 Sec.  7.11
```

$KEYIN FUNCTION $KEYIN

Workstation, Accept a String From Keyboard to Memory

```
Category: User Function Routine
Entered : 82 Jul 01                    Updated : 84 Jul 01   |*
File    : D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF   |*
                    if mode is other than 0)            |*

Syntax  : $KEYIN    (line, mode, &length, &hor, &ver)
Result  : D$CCODE

Entry 1 : line    ^ CHAR; Pointer to input area
Entry 2 : mode      BYTE; $WSIOMODE: $WSM....
Entry 3 : length    BYTE; Input area (1 for $WSENTK)
Entry 4 : hor       BYTE; Initial hor cur pos ($WSLC)
Entry 5 : ver       BYTE; Initial ver cur pos ($WSBL-x)
* Exit 3: length    BYTE; Initial length - keys input
* Exit 4: hor       BYTE; cursor position of $WSENTK
* Exit 5: ver       BYTE; cursor position of $WSENTK

If Error: IF $KEYIN (...) && D$CFLAG THEN $ERMSG( );
          $WSIO errors result in call to $ERMSG.
See Also: $GETLINE for error messages
          $WCONFIG for screen size.
          LISTS section: $WS... Workstation Keyboard   |*
          Codes                                        |*
SPRM Ref: KEYIN$       Vol.2 Sec.  7.3
```

Workstation-IF, Timeout Controlled Version of $KEYIN

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 84 Jul 01   |*
File    : D$UFRWS (also D$RMSWS for $WSIOMODE TYPDEF   |*
                   if mode is other than 0)               |*

Syntax  : $KEYINTO  (line, mode, &length, &hor, &ver)
Result  : D$CCODE

Entry 1 : line    ^ CHAR; Pointer to input area
Entry 2 : mode      BYTE; $WSIOMODE: $WSM....
Entry 3 : length    BYTE; Input area (1 for $WSENTK)
Entry 4 : hor       BYTE; Initial hor cur pos($WSLC)
Entry 5 : ver       BYTE; Initial ver cur pos($WSBL-x)
* Exit 3: length    BYTE; Initial length - keys input
* Exit 4: hor       BYTE; cursor position of $WSENTK
* Exit 5: ver       BYTE; cursor position of $WSENTK

If Error: IF $KEYINTO (...) && D$CFLAG THEN $ERMSG( );
          $WSIO errors result in call to $ERMSG.
See Also: $GETLINE for error messages
          $WCONFIG for screen size.
          LISTS section: $WS... Workstation Keyboard   |*
          Codes                                        |*
SPRM Ref: KEYINTO$      Vol.2 Sec.  7.6
```

Add a Member to a Library

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 27
File : D$UFRLIB

Syntax : **$LBADD** ***(pfdb, member, &dirmem, &lsn)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $LBADD *(pfdb,member,&dirmem,&lsn);***

Entry 1 :	***pfdb***	**^ $PFDB;** Opened PFDB
Entry 2 :	***member***	**^ $MEMBER;** New Member name
** Exit 3*:	***dirmem***	**^ $MEMBER;** Member entry in directory
** Exit 4*:	***lsn***	**UNSIGNED;** LSN of directory sector in disk buffer

If Error: **IF *varRslt* && D$CFLAG & *varRslt* && D$ZFLAG=0**
THEN *(duplicate member name);*
IF *varRslt* && D$CFLAG & *varRslt* && D$ZFLAG
THEN *(file format error or entry error);*
Remarks : Member source file must then be copied into library beginnig at LSN indicated in "dirmem.$LIBMLSN"
SPRM Ref: LBADD$ Vol.2 Sec. 9.5

$LBDEL FUNCTION $LBDEL

Delete a Member From a Library

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRLIB

Syntax : **$LBDEL** ***(pfdb, name)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB**; Opened PFDB
Entry 2 : ***name*** **^ $LNAMET**; member name to delete

If Error: **IF $LBDEL (...) && D$CFLAG**
THEN *(no such member);*
Remarks : The library entry will be marked *UNUSED*
SPRM Ref: LBDEL$ Vol.2 Sec. 9.4

$LBFIND FUNCTION $LBFIND

Locate Library Member

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 27
File : D$UFRLIB

Syntax : **$LBFIND** ***(pfdb, name, &member)***
Result : *D$CCODE*
Assign : ***varRslt*** := **$LBFIND** ***(pfdb, name, &member);***

Entry 1 : ***pfdb*** **^ $PFDB**; Opened library file
Entry 2 : ***name*** **^ $LNAMET**; member name
** Exit 3*: ***member*** **^ $MEMBER**; member entry in directory sector in zeroth buffer of PFDB

If Error: **IF *varRslt* && D$CFLAG & *varRslt* && D$ZFLAG=0**
THEN *(member not found)*
IF *varRslt* && D$CFLAG & *varRslt* && D$ZFLAG
THEN *(library file format or entry error)*
Remarks : PFDB values $PTODO,$PDONE,$PCLSN change as routine progresses.$PBUFL used for I/O.
SPRM Ref: LBFIND$ Vol.2 Sec. 9.1

Find the First Free Sector in a Library

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRLIB

Syntax : **$LBFREE** ***(pfdb, &lsn)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB;** Opened PFDB
** Exit 2*: ***lsn*** **UNSIGNED;** Highest USED LSN encounterd

If Error: **IF $LBFREE (...) && D$CFLAG**
THEN *(library file format error);*
SPRM Ref: LBFREE$ Vol.2 Sec. 9.3

Locate Library Member and Return LSN

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMS

Syntax : **$LBGTLSN** ***(type, pfdb, name, &lsn)***
Result : *D$CCODE*
Assign : ***varRslt*** **:= $LBGTLSN *(type,pfdb,name,&lsn);***
IF *varRslt* && D$CFLAG=0 & *varRslt* &&
D$ZFLAG=0 THEN *(member type mismatch);*
IF *varRslt* && D$CFLAG=0 & *varRslt* &&
D$ZFLAG THEN *(member type correct);*

Entry 1 : ***type*** **BYTE**; Member Type ($LIB...)
that end in : $LIBTYPE : D$RMSSTRUCT:
Entry 2 : ***pfdb*** **^ $PFDB**; same as $LBFIND
Entry 3 : ***name*** **^ $LNAMET**; same as $LBFIND
** Exit 4*: ***lsn*** **UNSIGNED**; LSN of Member

If Error: **IF *varRslt* && D$CFLAG & *varRslt* &&**
D$ZFLAG=0 THEN *(member not found);*
IF *varRslt* && D$CFLAG & *varRslt* &&
D$ZFLAG THEN *(library file format or entry error);*
See Also: $LIBTYPE in TYPES for more info on Entry 1
SPRM Ref: LBGTLSN$ Vol.2 Sec. 9.2

Load an Overlay

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMS

Syntax  : $LOAD     (pfdb, lsn, &start)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : lsn     UNSIGNED; LSN of abs header sector
* Exit 3: start   $STARTADR; Starting adr of overlay

If Error: IF  $LOAD (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$LOAD
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECLOAD2,     |*
                $ECLOAD3,$ECLOAD4,$ECSI05,$ECLOAD6,    |*
                $ECLOAD7,$ECLOAD8,$ECSI010             |*
SPRM Ref: $LOAD         Vol.4 Sec.  4.2
```

Invoking the Relocating Loader

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRRLD

Syntax : **$LOADREL** ***(work, param, &lastus, &sadr)***
Result : *D$CCODE*
Assign : ***varRslt*** := **$LOADREL*(work,param,&lastus,&sadr);***
IF *varRslt* && D$CFLAG=0 & *varRslt* &&
D$ZFLAG=0 THEN *(undefined symbol encountered, "lastus" points to last undefined symbol name);*
IF *varRslt* && D$CFLAG=0 & *varRslt* && D$ZFLAG
THEN *(all symbols defined);*

Entry 1 : ***work*** **^ [256] BYTE;** Loader work page
Entry 2 : ***param*** **^ $RLPARAM;** Loader parameter list
** Exit 3*: ***lastus*** **^ $RLNAME;** Loader name
** Exit 4*: ***sadr*** **$STARTADR;** Starting address or $NOADR

If Error: **IF *varRslt* && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SECR or $SECWAIT
$ERRC.$CODE = (see $SECR or $SECWAIT)
$ERRC.$FUNC = $UECLDR
$ERRC.$CODE = $UECLDR1,$UECLDR2,$UECLDR4,
$UECLDR5
SPRM Ref: LOADREL$ Vol.2 Sec. 11.2

$LOCKFAV FUNCTION $LOCKFAV

Lock / Unlock a Specified FAV

Category: System Call |
Entered : 82 Dec 01 *Updated* : 84 Jul 25 |
File : D$RMSSPEC |

Syntax : **$LOCKFAV** ***(mode, pfdb)***
Result : *D$CCODE*

Entry 1 : ***mode*** BYTE; $LFLOKSP or $LFULOKS
Entry 2 : ***pfdb*** **^ $PFDB;**

If Error: **IF $LOCKFAV (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$LOCKFAV |
$ERRC.$CODE = $ECUMAV,$ECSI001,$ECLKF2 |

$LOCKRIM FUNCTION $LOCKRIM

RIM Lockout, Attempt to Open Pipe ?(on hold)

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRSYS

Syntax : **$LOCKRIM** ***(name, pfdb)***
Result : *D$CCODE*
Assign : ***varRslt*** := **$LOCKRIM** ***(name, pfdb);***
IF ***varRslt*** **&& D$ZFLAG THEN** *(Lock Failed);*

Entry 1 : ***name*** **^ $NAMET;** RIM Name
Entry 2 : ***pfdb*** **^ $PFDB;** To be used for pipe

If Error: **IF** ***varRslt*** **&& D$CFLAG THEN $ERMSG ();**
Remarks : Failure means someone is using RIM in DOS. Pipe name will be same as RIM with an "X" appended (a 12th letter in $NAMET would be discarded). Pipe password is 1st 8 chars.
See Also: $UNLKRIM for releasing RIM from Pipe
SPRM Ref: $LOCKRIM Vol.2 Sec. 5.6.1

$LOGCLR FUNCTION $LOGCLR

Workstation-IF, Clear Logging Flags

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 10
File    : D$UFRWS

Syntax  : $LOGCLR   (flag, &old)
Result  : none

Entry 1 : flag    BYTE; flag bits to be cleared ($PCRLF..)
* Exit 2: old     BYTE; flag bits previously set($PCRLF..)

If Error: No Error Occurs
See Also: $LOGSET for log flag bits; type PCR, $PCRLOGF
          for interpretation of bits
SPRM Ref: LOGCLR$        Vol.2 Sec.  7.19
```

$LOGGING FUNCTION $LOGGING

Workstation-IF, Determine if Logging is Active

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 82 Dec 01
File    : D$UFRWS

Syntax  : $LOGGING  ()
Result  : D$CCODE
Assign  : varRslt := $LOGGING  ();
          IF varRslt && D$CFLAG=0 THEN (LOG not Active);
          IF varRslt && D$CFLAG THEN (LOG  Active);

If Error: No Error Occurs
SPRM Ref: LOGGING$       Vol.2 Sec.  7.21
```

$LOGSET FUNCTION $LOGSET

Workstation-IF, Set Logging Flags

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRWS

Syntax : $LOGSET *(flag, &old)*
Result : *none*

Entry 1 : ***flag*** BYTE; bits to be set ($PCRLF..)
** Exit 2*: ***old*** BYTE; previous flag bits set ($PCRLF..)

If Error: **No Error Occurs**
See Also: type PCR, $PCRLOGF for interpretation of bits
Remarks : Log Flag Bits: $PCRLFAC, $PCRLFSP, $PCRLFEO, $PCRLFNI, $PCRLFFO, $PCRLFHR
SPRM Ref: LOGSET$ Vol.2 Sec. 7.18

LOOP FUNCTION LOOP

Execute Substatements Until WHILE Expression = 0

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax :
```
LOOP {statement; [...statement;]
WHILE expression; [..statement]};
also:
LOOP {
   [...statement;]
 WHILE expression;
   statement;
   [...statement;]
   };
```

Remarks : LOOP WHILES may be nested; they may be one of the *statements* shown in the syntax. The order of expression and statement execution is controlable by placement in the syntax structure.
DASL Doc: 73,74 March 1982

$MAP4K FUNCTION $MAP4K

System-IF, Allocate Memory For a PFDB

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 01
File : D$UFRSYS

Syntax : **$MAP4K** ***(pfdb)***
Result : D$CCODE |*

Entry 1 : ***pfdb*** **^ $PFDB;**

If Error: **If $MAP4K (...) && D$CFLAG THEN $ERMSG( );** |*
SPRM Ref: MAP4K$ Vol.2 Sec. 5.5

$MEMCTL FUNCTION $MEMCTL

Memory Control

Category: System Call
Entered : 83 Apr 02 *Updated* : 83 Jul 19
File : D$RMSMEM

Syntax : **$MEMCTL** ***(func)***
Result : *D$CCODE*

Entry 1 : ***func*** **BYTE;** memory control codes ($MC...)
If Error: **IF $MEMCTL (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$MEMCTL
$ERRC.$CODE = $ECUMAV,$ECMCTL1,$ECMCTL2,
$ECMCTL3
Remarks : Memory control codes: $MCMDON,$MCMDOFF,
$MCMDTST,$MCDSON,$MCDSOFF,$MCDSTST

$MEMGET FUNCTION $MEMGET

Obtain PSK for Available User Memory Sector

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSMEM

Syntax : **$MEMGET** ***(&psk)***
Result : *D$CCODE*

* *Exit 1*: ***psk*** **BYTE;** PSK to use in $MEMMAP

If Error: **IF $MEMGET (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$MEMGET
$ERRC.$CODE = $ECMGET0, $ECMGET1 |
See Also: $$MEMGET for User Function Routine version
SPRM Ref: $MEMGET Vol.4 Sec. 5.2.1

$$MEMGET FUNCTION $$MEMGET

System-IF, Obtain A Physical Memory Sector

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRSYS

Syntax : **$$MEMGET** ***(&psk)***
Result : *D$CCODE*

* *Exit 1*: ***psk*** **BYTE;** Next available sector PSK

If Error: **IF $$MEMGET (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC=SC$MEMGET
$ERRC.$CODE=$ECMGET0
$ERRC.$FUNC=$UECMEM partition limit exceeded
Remarks : Checks partition memory limit in PCR
See Also: $MEMGET for System Call version
SPRM Ref: MEMGET$ Vol.2 Sec. 5.3

Obtain PSK of a Mapped Memory Sector

```
Category: System Call
Entered : 82 Jul 01                Updated : 84 Jul 25  |*
File    : D$RMSMEM

Syntax  : $MEMKEY   (msn, &psk)
Result  : D$CCODE
Assign  : varRslt := $MEMKEY (msn, &psk);
          IF varRslt && D$CFLAG=0 & varRslt &&
          D$ZFLAG THEN (write access);
          IF varRslt && D$CFLAG=0 & varRslt &&
          D$ZFLAG=0 THEN (read only access);
          IF varRslt && D$CFLAG & $ERRC.$CODE
          = $ECMKEY0 THEN (MSN is greater than avail)
          or $ERMSG() ;
          IF varRslt && D$CFLAG & $ERRC.$CODE
          = $ECMKEY1 THEN (MSN not currently mapped)
          or $ERMSG() ;

Entry 1 : msn     BYTE; MSN Index: 0-15 << 4 and
                        low order bit is ON/OFF
* Exit 2: psk     BYTE; PSK mapped in given MSN

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$MEMKEY
          $ERRC.$CODE = $ECMKEY0,$ECMMAP1 (see Assign:) |*
SPRM Ref: $MEMKEY         Vol.4 Sec.  5.2.4
```

Map a Physical Memory Sector into Logical Space

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 01
File    : D$RMSMEM

Syntax  : $MEMMAP   (msn, psk)
Result  : D$CCODE
Assign  : varRslt := $MEMMAP (msn, psk);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (write access);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN (read only access);
          IF varRslt && D$CFLAG & $ERRC.$CODE = $ECMMAP1
          THEN (MSN is greater than avail.) or $ERMSG( );

Entry 1 : msn  BYTE; MSN Logical Index: see $MEMKEY
Entry 2 : psk  BYTE; PSK obtained from $MEMGET,$MEMKEY

If Error: IF varRslt && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = SC$MEMMAP
          $ERRC.$CODE = $ECMMAP0,$ECMMAP1 (see ASSIGN)
Remarks : The User Task can map a sector instead of PCR
          in MSN 0. Nucleus restores PCR on exit to CI.
SPRM Ref: $MEMMAP       Vol.4 Sec.  5.2.3
```

Change Memory Protection

Category: System Call
Entered : 83 Apr 02 *Updated* : 84 Jul 25 |*
File : D$RMSMEM

Syntax : **$MEMPROT** ***(prot, psk)***
Result : *D$CCODE*

Entry 1 : ***prot*** **BYTE;** Mode $MPROTRO, $MPROTRW
Entry 2 : ***psk*** **BYTE;**

```
If Error: IF  $MEMPROT (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$MEMPROT
          $ERRC.$CODE = $ECMREL0,$ECMREL1,$ECMREL2,    |*
                        $ECMREL3                       |*
```

$MEMREL FUNCTION $MEMREL

Release a Memory Sector

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |
File : D$RMSMEM

Syntax : **$MEMREL** ***(psk)***
Result : *D$CCODE*

Entry 1 : ***psk*** **BYTE;** PSK of 4k bytes to release
(..But not the PCR)

If Error: **IF $MEMREL (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$MEMREL
$ERRC.$CODE = $ECMREL0,$ECMREL1,$ECMREL2, |
$ECMREL3 |
See Also: $$MEMREL for User Function Routine version
SPRM Ref: $MEMREL Vol.4 Sec. 5.2.2

$$MEMREL FUNCTION $$MEMREL

System-IF, Release a Physical Memory Sector

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRSYS

Syntax : **$$MEMREL** ***(psk)***
Result : *D$CCODE*

Entry 1 : ***psk*** **BYTE;** PSK of sector to be released

If Error: **IF $$MEMREL (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$MEMREL
$ERRC.$CODE = $ECMREL0,$ECMREL1,$ECMREL2
See Also: $MEMREL for System Call version of routine
SPRM Ref: MEMREL$ Vol.2 Sec. 5.4

MFREMEM$ FUNCTION MFREMEM$

Name of EXTERNAL RMS UFR used by $MMFREMEM

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

MGETCLR$ FUNCTION MGETCLR$

Name of EXTERNAL RMS UFR used by $MMGETCLR

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

MGETFST$ FUNCTION MGETFST$

Name of EXTERNAL RMS UFR used by $MMGETFST

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

MGETMEM$ FUNCTION MGETMEM$

Name of EXTERNAL RMS UFR used by $MMGETMEM

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

MGETPAG$ FUNCTION MGETPAG$

Name of EXTERNAL RMS UFR used by $MMGETPAG

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Numeric, Unsigned 24-bit Multiplication

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 0
File : D$UFRNUM

Syntax : **$MLTPLY3** ***(val1, val2, &result)***
Result : *D$CCODE*

Entry 1 : ***val1*** **ULONG;**
Entry 2 : ***val2*** **ULONG;**
** Exit 3*: ***result*** **ULONG;**

If Error: **IF $MLTPLY3 (...) && D$CFLAG THEN overflo**
SPRM Ref: MLTPLY3$ Vol.2 Sec. 4.14

Deallocate a Block of Managed Memory

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 1
File : D$UFRMEM

Syntax : **$MMFREMEM** ***(block)***
Result : *D$CCODE*

Entry 1 : ***block*** **^ BYTE;** First user byte of block

If Error: **IF $MMFREMEM (...) && D$CFLAG**
THEN *(invalid block address passed);*
See Also: $MMINIT for remarks and list
of related functions
SPRM Ref: MFREMEM$ Vol.2 Sec. 12.2.4

$MMGETCLR FUNCTION $MMGETCLR

Allocate a Cleared Block of Managed Memory

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRMEM

Syntax : **$MMGETCLR** ***(length, &block)***
Result : *D$CCODE*

Entry 1 : ***length*** **UNSIGNED;** Length of block desired
** Exit 2*: ***block*** **^ BYTE;** first byte of block allocated

If Error: **IF $MMGETCLR (...) && D$CFLAG**
THEN *(memory not available);*
Remarks : Identical to $MMGETMEM excepts clears allocated area to binary zeros.
See Also: $MMINIT for remarks and list of related functions
SPRM Ref: MGETCLR$ Vol.2 Sec. 12.2.2

$MMGETFST FUNCTION $MMGETFST

Allocate Free Space Block of Managed Memory

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRMEM

Syntax : **$MMGETFST** ***(length, &block)***
Result : *D$CCODE*

Entry 1 : ***length*** **UNSIGNED;** block length desired
** Exit 2*: ***block*** **^ BYTE;** first by of block allocated

If Error: **IF $MMGETFST (...) && D$CFLAG**
THEN *(memory not available);*
Remarks : Allocation by extending unused space; similar to $MMGETMEM but does not search free list.
See Also: $MMINIT for remarks and list of related functions
SPRM Ref: MGETFST$ Vol.2 Sec. 12.2.3

$MMGETMEM FUNCTION $MMGETMEM

Allocate a Block of Managed Memory from Free List

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRMEM

Syntax : **$MMGETMEM** ***(length, &block)***
Result : *D$CCODE*

Entry 1 : ***length*** **UNSIGNED;** Block length desired
* *Exit 2*: ***block*** **^ BYTE;** first byte of block allocated

If Error: **IF $MMGETMEM (...) && D$CFLAG**
THEN *(memory not available);*
See Also: $MMINIT for remarks and list of related functions
SPRM Ref: MGETMEM$ Vol.2 Sec. 12.2.1

$MMGETPAG FUNCTION $MMGETPAG

Obtain a Page of Logical Managed Memory Space

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRMEM

Syntax : **$MMGETPAG** ***(&msb)***
Result : *D$CCODE*

* *Exit 1*: ***msb*** **BYTE;** MSB of allocated Page address

If Error: **IF $MMGETPAG (...) && D$CFLAG**
THEN *(no memory available);*
Remarks : Pages are allocated from highest address of logical managed memory space downwards.
See Also: $MMINIT for remarks and list of related functions
SPRM Ref: MGETPAG$ Vol.2 Sec. 12.3.1

MMGINIT$ FUNCTION MMGINIT$

Name of EXTERNAL RMS UFR used by $MMINIT

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

$MMINIT FUNCTION $MMINIT

Initialize Management of Already Mapped Memory

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRMEM

Syntax : **$MMINIT** *(top)*
Result : *none*

Entry 1 : ***top*** ^ BYTE; 1st byte,managed logical space

If Error: **No Error Occurs**
Remarks : Initializes management of variable size blks or 256-byte pages of memory previously allocated and mapped in (4k sectors).
See Also: $MMFREMEM,$MMGETCLR,$MMGETFST,$MMGETMEM, $MMGETPAG,$MMRETPAG
SPRM Ref: MMGINIT$ Vol.2 Sec. 12.1

$MMRETPAG FUNCTION $MMRETPAG

Release a Page of Logical Managed-Memory Space

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRMEM

Syntax : **$MMRETPAG** ***(msb)***
Result : *D$CCODE*

Entry 1 : ***msb*** **BYTE;** MSB of page previously allocated.

If Error: **IF $MMRETPAG (...) && D$CFLAG**
THEN *(invalid MSB passed);*
See Also: $MMINIT for remarks and list
of related functions
SPRM Ref: MRETPAG$ Vol.2 Sec. 12.3.2

MRETPAG$ FUNCTION MRETPAG$

Name of EXTERNAL RMS UFR used by $MMRETPAG

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Err-Msg, Locate a Message and Copy Into $MSG

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 84 Aug 08  |*
File    : D$RMS

Syntax  : $MSGC  (errNum, &length)
Result  : D$CCODE

Entry 1 : errNum  UNSIGNED; user defined msg. number     |*
* Exit 2: length    BYTE;

If Error: IF  $MSGC (...) && D$CFLAG THEN $ERMSG ();
          $ERRC will contain $SECR or $SECWAIT error
          codes indicate Command File I/O error.
Remarks : Opens the program command file library from   |*
          information in $PCRCMDN and $PCRCMDE to        |*
          access user defined messages in the MESSAGE    |*
          member.                                        |*
          $MSGCGET is called following the open.         |*
          Use $MSGCXO to open members with a name        |*
          other than MESSAGE.                            |*
See Also: $MSGCGET for alternate routine
SPRM Ref: MSGC$         Vol.2 Sec.  3.3.1
```

$MSGCGET FUNCTION $MSGCGET

Err-Msg, Locate and Deliver a Message

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |
File : D$UFRERR

Syntax : **$MSGCGET** ***(pfdb, lsn, message, &length)***
Result : *D$CCODE*

```
Entry 1 : pfdb    ^ $PFDB; the command file                  |
Entry 2 : lsn       UNSIGNED;of member (from $LIBINFO)       |
Entry 3 : message   UNSIGNED; message code number            |
* Exit 4: length    BYTE; IF > 0 message found,
                          IF = 0 message not found
```

If Error: **IF $MSGCGET (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC will contain $SECR or $SECWAIT error codes indicate Command File I/O error.
Remarks : Smaller than $MSGC and contains no internal calls to other UFR's. Command file must be open, and message member located
See Also: $MSGC for alternative routine which opens for itself.
SPRM Ref: MSGCGET$ Vol.2 Sec. 3.3.2

$MSGCXO FUNCTION $MSGCXO

Open, Position to a Command Lib MESSAGE Member

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |
File : D$UFRERR

Syntax : **$MSGCXO** ***(name)***
Result : *D$CCODE*

Entry 1 : ***name*** **^ $LNAMET**; member name |

If Error: **IF $MSGCXO (...) && D$CFLAG**
THEN *(file / member not found);*
SPRM Ref: MSGCXO$ Vol.2 Sec. 3.3.3

$NEWPCR FUNCTION $NEWPCR

Environment, Create New PCR for Independent Task

```
Category: User Function Routine
Entered : 82 Jul 01                  Updated : 83 Jul 30
File    : D$UFRENV

Syntax  : $NEWPCR  (msn, commands, length)
Result  : D$CCODE

Entry 1 : msn          BYTE; MSN to be used for new PCR
Entry 2 : commands ^ CHAR; Adr of new command stack
Entry 3 : length       UNSIGNED; length of new cmd stack

If Error: IF  $NEWPCR (...) && D$CFLAG THEN $ERMSG ();
          Command stack overflow
SPRM Ref: NEWPCR$       Vol.2 Sec.  2.15
```

$NQDQBLD FUNCTION $NQDQBLD

NQDQ, Build a Message Block

```
Category: User Function Routine
Entered : 82 Jul 01                  Updated : 83 Apr 02
File    : D$UFRNQDQ

Syntax  : $NQDQBLD  (list, msg, &err)
Result  : D$CCODE

Entry 1 : list ^ ^ $OPENPT; adr of adr list, 22 max
Entry 2 : msg    ^ $NQDQMSG; address of output area
* Exit 3: err      BYTE;

If Error: IF  $NQDQBLD (...) && D$CFLAG
          THEN ("err" is error code);
          err = 14  too many items in request
          err = 15  error during $INFO
          $ERRC.$FUNC = SC$INFO
          $ERRC.$CODE = see $INFO
SPRM Ref: NQDQBLD$      Vol.2 Sec. 10.3
```

NQDQ, Check Limited Request

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 84 Jul 01
File    : D$UFRNQDQ

Syntax  : $NQDQCHK  (&err)
Result  : D$CCODE
Assign  : varRslt := $NQDQCHK  (&err);
          IF varRslt && D$CFLAG && ("err" = 0)
          THEN (request pending..NO ERROR);
          IF varRslt && D$CFLAG=0
          THEN (request granted);

* Exit 1: err     BYTE;

If Error: IF varRslt && D$CFLAG
          THEN ("err" is error code);
          err = 0  request pending..NO ERROR
                5  invalid request code
                6  unrecognizable response received
                7  user task not logged on
                8  no limited enqueue active
               21  error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT
          $ERRC.$CODE = see $SECWAIT
Remarks : Use with $NQDQENL, to see if limited enqueue
          granted.
SPRM Ref: NQDQCHK$      Vol.2 Sec. 10.6
```

NQDQ, Request a Limited Enqueue

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 10
File    : D$UFRNQDQ

Syntax  : $NQDQENL  (msg, &err)
Result  : D$CCODE

Entry 1 : msg  ^ $NQDQMSG; pointer to message block
* Exit 2: err    BYTE;

If Error: IF  $NQDQENL (...) && D$CFLAG
          THEN ("err" is error code);
          err = 5  invalid request code
                6  unrecognizable response received
               11  error during $SECR to UCP
               16  error during $SECW to RSP
               21  error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT, SC$SECW, SC$SECR
          $ERRC.$CODE = see $SECWAIT,$SECW or $SECR
Remarks : Same as $NQDQENQ except control returned
          immediately to calling user. Then user must
          use $NQDQCHK to determine in request was
          granted.
See Also: $NQDQENQ for differences; $NQDQCHK for
          results
SPRM Ref: NQDQENL$      Vol.2 Sec. 10.5
```

NQDQ, Enqueue on a Resource

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 May 03
File    : D$UFRNQDQ

Syntax  : $NQDQENQ  (msg, &err)
Result  : D$CCODE

Entry 1 : msg   ^ $NQDQMSG; pointer to message block
* Exit 2: err     BYTE;

If Error: IF  $NQDQENQ (...) && D$CFLAG
          THEN ("err" is error code);
          err = 5  invalid request code
                6  unrecognizable response received
                7  user task not logged on
               11  error during $SECR to UCP
               16  error during $SECW to RSP
               21  error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT, SC$SECW, SC$SECR
          $ERRC.$CODE = see $SECWAIT,$SECW or $SECR
SPRM Ref: NQDQENQ$      Vol.2 Sec. 10.4
```

NQDQ, Disconnect from the System

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$UFRNQDQ

Syntax  : $NQDQLGF  (&err)
Result  : D$CCODE

* Exit 1: err    BYTE;

If Error: IF  $NQDQLGF (...) && D$CFLAG
          THEN ("err"=error code);
          err = 7 user task not logged on
               16 error during $SECW to RSP
               17 error during $CLOSE
          $ERRC.$FUNC = SC$SECW or SC$CLOSE
          $ERRC.$CODE = see $SECW or $CLOSE
SPRM Ref: NQDQLGF$      Vol.2 Sec. 10.2
```

NQDQ, Connect to the System

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 May 03
File    : D$UFRNQDQ

Syntax  : $NQDQLGN (logon, &err)
Result  : D$CCODE

Entry 1 : logon  ^ $NAMEEXTENV; Logon parameter table
* Exit 2: err      BYTE;

If Error: IF  $NQDQLGN (...) && D$CFLAG
          THEN (error code in "err");
          err = 1 unable to open CLP
                2 controller not allowing logons
                3 user task already logged on
                4 version/modification level mismatch
                5 invalid request code
                6 unrecognizable response received
               11 error during $SECR to UCP
               12 controller environment not found
               16 error during $SECW to RSP
               17 error during $CLOSE
               18 unable to open UCP
               19 unable to open RSP
               20 unable to read CLP
               21 error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT, SC$SECW, SC$SECR
          $ERRC.$CODE = see $SECWAIT,$SECW or $SECR
SPRM Ref: NQDQLGN$     Vol.2 Sec. 10.1
```

NQDQ, Release a Resource

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$UFRNQDQ

Syntax : **$NQDQREL** ***(&err)***
Result : *D$CCODE*

** Exit 1*: ***err*** **BYTE;**

If Error: IF **$NQDQREL (...) && D$CFLAG**
THEN *("err" is error code);*

```
err = 5 invalid request code
      6 unrecognizable response received
      7 user task not logged on
     11 error during $SECR to UCP
     16 error during $SECW to RSP
     21 error during $SECWAIT to UCP
$ERRC.$FUNC = SC$SECWAIT, SC$SECW, SC$SECR
$ERRC.$CODE = see $SECWAIT,$SECW or $SECR
```

SPRM Ref: NQDQREL$ Vol.2 Sec. 10.9

NQDQ, Reset the Controller 4-byte Counters

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$UFRNQDQ

Syntax : **$NQDQRST** ***(&err)***
Result : *D$CCODE*

* *Exit 1*: ***err*** **BYTE;**

```
If Error: IF  $NQDQRST (...) && D$CFLAG
          THEN ("err" is error code);
          err = 7  user task not logged on
               16  error during $SECW to RSP
          $ERRC.$FUNC = SC$SECW
          $ERRC.$CODE = see $SECW
Remarks : Resets counters $NQDQSTAT.CNTENQ
          thru $NQDQSTAT.CNTTO
SPRM Ref: NQDQRST$      Vol.2 Sec. 10.11
```

NQDQ, Acquire Controller Statistics

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 May 03
File    : D$UFRNQDQ

Syntax  : $NQDQSTA  (stat, &err)
Result  : D$CCODE

Entry 1 : stat   ^ $NQDQSTAT; counters
* Exit 2: err      BYTE;

If Error: IF  $NQDQSTA (...) && D$CFLAG
          THEN ("err" is error code);
          err = 5 invalid request code
                6 unrecognizable response received
                7 user task not logged on
               11 error during $SECR to UCP
               16 error during $SECW to RSP
               21 error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT or SC$SECW or SC$SECR
          $ERRC.$CODE = see $SECWAIT,$SECW or $SECR
SPRM Ref: NQDQSTA$      Vol.2 Sec.  10.10
```

NQDQ, Terminate an In-Progress Limited Enqueue

```
Category: User Function Routine
Entered : 82 Jul 01                  Updated : 83 May 03
File    : D$UFRNQDQ

Syntax  : $NQDQSTP  (&err)
Result  : D$CCODE

* Exit 1: err    BYTE;

If Error: IF  $NQDQSTP (...) && D$CFLAG
          THEN ("err" is error code);
          err = 5 invalid request code
                6 unrecognizable response received
                7 user task not logged on
               10 no limited enqueue active
               11 error during $SECR to UCP
               16 error during $SECW to RSP
               21 error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT, SC$SECW, SC$SECR
          $ERRC.$CODE = see $SECWAIT,$SECW or $SECR
SPRM Ref: NQDQSTP$      Vol.2 Sec. 10.8
```

NQDQ, Wait for a Limited Enqueue Request

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Apr 02
File    : D$UFRNQDQ

Syntax  : $NQDQWAT  (&err)
Result  : D$CCODE

* Exit 1: err    BYTE;

If Error: IF  $NQDQWAT (...) && D$CFLAG
          THEN ("err" is error code);
          err = 5 invalid request code
                6 unrecognizable response received
                7 user task not logged on
                9 no limited enqueue active
               21 error during $SECWAIT to UCP
          $ERRC.$FUNC = SC$SECWAIT
          $ERRC.$CODE = see $SECWAIT
SPRM Ref: NQDQWAT$      Vol.2 Sec. 10.7
```

Open a File; Search for ENV, which is not specified

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$RMS

Syntax  : $OPEN (mode, openpt)
Result  : D$CCODE

Entry 1 : mode      BYTE; open modes ($OM...)
Entry 2 : openpt ^ $OPENPT;

If Error: IF  $OPEN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$OPENENV
          $ERRC.$CODE = see $OPENENV error codes
          $ERRC.$FUNC = $UECOPN
          $ERRC.$CODE = $UECOPN0,$UECOPN1,$UECOPN2,
                        $UECOPN3
See Also: $OPENENV for Open Modes, Access Codes,
          Formats
SPRM Ref: $OPEN         Vol.2 Sec.  5.1
```

Open a File with Specified ENV

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$RMSIO

Syntax : **$OPENENV** *(mode, openpt)*
Result : *D$CCODE*

Entry 1 : ***mode*** **BYTE; $OM...**
Entry 2 : ***openpt*** **^ $OPENPT;** See $OPENPTS,special modes

```
If Error: IF  $OPENENV (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$OPENENV
          $ERRC.$CODE = All resources                       |*
                $ECUMAV,$ECSI005,$ECSI007,$ECSI021,         |*
                $ECSI023,$ECSI025,$ECSI028,$ECSI030,        |*
                $ECSI031,$ECSI034,$ECSI063                  |*

                        Disk resources                      |*
                $ECSI012,$ECSI013,$ECSI014,$ECSI015,        |*
                $ECSI016,$ECSI022,$ECSI024,$ECSI026,        |*
                $ECSI027,$ECSI045,$ECSI053,$ECSI055,        |*
                $ECSI064                                    |*

                        Pipe resources                      |*
                      $ECSI045,$ECSI055                     |*

                        Printer resources                   |*
                $ECSI029,$ECSI043,$ECSI045,$ECSI055         |*

                        Tape resources                      |*
                      $ECSI029,$ECSI043                     |*

                        Card reader resources               |*
                            $ECSI054                        |*

                        Comm resources                      |*
                $ECSI029,$ECSI043,$ECSI054,$ECSI057,        |*
                $ECSI059                                    |*
```

... *continued*

Remarks : Open Modes.:
$OMREAD,$OMEXCL,$OMSHARE,$OMCREAT,$OMPREP
Special.:
$OMCHECK,$OMREPAR (use $OPENPTS)

Access Codes.:
$ACATALG,$ACKILL,$ACREAD,$ACREATE,$ACRENM,
$ACREPX,$ACSECQ,$ACWRIT,$ACMAX

File Formats.: $FFMT...

SPRM Ref:	$OPENENV	Vol.4 Sec. 7.4.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 8.4.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 9.1.1.3
SPRM Ref:	$OPENENV	Vol.4 Sec. 9.2.1.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 9.3.1.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 9.4.1.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 10.1.1
SPRM Ref:	$OPENENV	Vol.4 Sec. 11.1

$PAKPW FUNCTION $PAKPW

Environment, Pack ACSII String Into Password

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRENV

Syntax : **$PAKPW** ***(uncomp, comp, &invalid)***
Result : ***D$CCODE***
Assign : ***varRslt*** **:= $PAKPW** ***(uncomp, comp, &invalid);***
IF ***varRslt*** **&& D$CFLAG=0 &** ***varRslt*** **&&**
D$ZFLAG=0 THEN ***(good password);***
IF ***varRslt*** **&& D$CFLAG=0 &** ***varRslt*** **&& D$ZFLAG**
THEN ***(null password);***

Entry 1 :	***uncomp***	**^ $UNPACKPW;**
Entry 2 :	***comp***	**^ $PACKPW;**
** Exit 3*:	***invalid***	**^ CHAR;**

If Error: **IF** ***varRslt*** **&& D$CFLAG THEN invalid password**
SPRM Ref: PAKPW$ Vol.2 Sec. 2.10

$PIPEGEN FUNCTION $PIPEGEN

Create a Pipe Resource

```
Category: System Call
Entered : 82 Jul 01              Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $PIPEGEN  (ptab, autoDel)
Result  : D$CCODE

Entry 1 : ptab     ^ $PIPEGENPT;
Entry 2 : autoDel   BOOLEAN; Alternate mode if true

If Error: IF  $PIPEGEN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$PIPEGEN
          $ERRC.$CODE = $ECUMAV,$ECSI028,$ECSI030,
                $ECSI031,$ECSI037,$ECSI040,$ECSI049    |*
SPRM Ref: $PIPEGEN      Vol.4 Sec.  9.1.1.1
```

$PIPEUSE FUNCTION $PIPEUSE

Check Local Pipe-in-use Status

```
Category: System Call
Entered : 82 Jul 01              Updated : 82 Dec 01
File    : D$RMSIO

Syntax  : $PIPEUSE  (pfdb)
Result  : D$CCODE
Assign  : varRslt := $PIPEUSE  (pfdb);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (no other task has pipe opened.);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN another task has pipe opened.

Entry 1 : pfdb  ^ $PFDB;

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$PIPEUSE
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,
                        $ECSI007,$ECSI010,$ECSI025
SPRM Ref: $PIPEUSE      Vol.4 Sec.  9.1.1.2
```

$PUTELGX FUNCTION $PUTELGX

Workstation-IF, Log Message

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul
File    : D$UFRWS

Syntax  : $PUTELGX  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTELGX (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTELGX$      Vol.2 Sec.  7.1
```

$PUTELOG FUNCTION $PUTELOG

Workstation-IF, Home-Down Roll, Log Error Message

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTELOG  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTELOG (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTELOG$      Vol.2 Sec.  7.1
```

$PUTERP FUNCTION $PUTERP

Workstation, Home-Down Roll, Log/Display $MSG, Error

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS

Syntax : **$PUTERP** ***(&hor, &ver)***
Result : *D$CCODE*

Entry 1 : **hor** BYTE; Initial hor curs pos($WSLC)
Entry 2 : **ver** BYTE; Initial ver curs pos($WSBL-x)
** Exit 1*: **hor** BYTE; Ending cursor position
** Exit 2*: **ver** BYTE; Ending cursor position

If Error: **IF $PUTERP (...) && D$CFLAG THEN $ERMSG ();**
Remarks : Entry requires $WSIO mode bits in PUTWSMD$ (usually 0's). Message is in $MSG. $WCONFIG for screen size.
SPRM Ref: PUTERP$ Vol.2 Sec. 7.1

$PUTERPX FUNCTION $PUTERPX

Workstation, Log/Display from $MSG, Error

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS

Syntax : **$PUTERPX** ***(&hor, &ver)***
Result : *D$CCODE*

Entry 1 : **hor** BYTE; Initial hor curs pos($WSLC)
Entry 2 : **ver** BYTE; Initial ver curs pos($WSBL-x)
** Exit 1*: **hor** BYTE; Ending cursor position
** Exit 2*: **ver** BYTE; Ending cursor position

If Error: **IF $PUTERPX (...) && D$CFLAG THEN $ERMSG ();**
Remarks : Entry requires $WSIO mode bits in PUTWSMD$ (usually 0's). Message is in $MSG externally defined buffer. $WCONFIG for screen size.
SPRM Ref: PUTERPX$ Vol.2 Sec. 7.1

$PUTERRR FUNCTION $PUTERRR

Workstation, Home-Down Roll, Log/Display, Error

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTERRR  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTERRR (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTERRR$     Vol.2 Sec.  7.1
```

$PUTERRX FUNCTION $PUTERRX

Workstation, Log/Display Message, Error

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTERRX  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTERRX (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTERRX$     Vol.2 Sec.  7.1
```

$PUTLINE FUNCTION $PUTLINE

Workstation, Home-Down Roll, Display Message

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTLINE  (message, &hor,&ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTLINE (...) && D$CFLAG THEN $ERMSG ( );
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTLINE$     Vol.2 Sec.  7.1
```

$PUTLINX FUNCTION $PUTLINX

Workstation, Display Message

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTLINX  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTLINX (...) && D$CFLAG THEN $ERMSG ( );
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTLINX$     Vol.2 Sec.  7.1
```

$PUTLNP FUNCTION $PUTLNP

Workstation, Home-Down Roll, Display from $MSG

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS

Syntax : **$PUTLNP** ***(&hor, &ver)***
Result : *D$CCODE*

Entry 1 :	***hor***	BYTE; Initial hor curs pos($WSLC)
Entry 2 :	***ver***	BYTE; Initial ver curs pos($WSBL-x)
* *Exit 1*:	***hor***	BYTE; Ending cursor position
* *Exit 2*:	***ver***	BYTE; Ending cursor position

If Error: **IF $PUTLNP (...) && D$CFLAG THEN $ERMSG ();**
Remarks : Entry requires $WSIO mode bits in PUTWSMD$ (usually 0's). Message is in $MSG externally defined buffer. $WCONFIG for screen size.
SPRM Ref: PUTLNP$ Vol.2 Sec. 7.1

$PUTLNPX FUNCTION $PUTLNPX

Workstation, Display from $MSG

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$UFRWS

Syntax : **$PUTLNPX** ***(&hor, &ver)***
Result : *D$CCODE*

Entry 1 :	***hor***	BYTE; Initial hor curs pos($WSLC)
Entry 2 :	***ver***	BYTE; Initial ver curs pos($WSBL-x)
* *Exit 1*:	***hor***	BYTE; Ending cursor position
* *Exit 2*:	***ver***	BYTE; Ending cursor position

If Error: **IF $PUTLNPX (...) && D$CFLAG THEN $ERMSG ();**
Remarks : Entry requires $WSIO mode bits in PUTWSMD$ (usually 0's). Message is in $MSG externally defined buffer. $WCONFIG for screen size.
SPRM Ref: PUTLNPX$ Vol.2 Sec. 7.1

$PUTLOG FUNCTION $PUTLOG

Workstation, Home-Down Roll, Log Message

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTLOG   (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTLOG (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTLOG$       Vol.2 Sec.  7.1
```

$PUTLOGX FUNCTION $PUTLOGX

Workstation, Log Message

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTLOGX  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTLOGX (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTLOGX$      Vol.2 Sec.  7.1
```

$PUTNOP FUNCTION $PUTNOP

Workstation, Home-Down Roll, Log/Display from $MSG

```
Category: User Function Routine
Entered : 82 Jul 01                    Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTNOP   (&hor, &ver)
Result  : D$CCODE

Entry 1 : hor    BYTE; Initial hor curs pos($WSLC)
Entry 2 : ver    BYTE; Initial ver curs pos($WSBL-x)
* Exit 1: hor    BYTE; Ending cursor position
* Exit 2: ver    BYTE; Ending cursor position

If Error: IF  $PUTNOP (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). Message is in $MSG externally
          defined buffer. $WCONFIG for screen size.
SPRM Ref: PUTNOP$        Vol.2 Sec.  7.1
```

$PUTNOPX FUNCTION $PUTNOPX

Workstation, Log/Display from $MSG

```
Category: User Function Routine
Entered : 82 Jul 01                    Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTNOPX (&hor, &ver)
Result  : D$CCODE

Entry 1 : hor    BYTE; Initial hor curs pos($WSLC)
Entry 2 : ver    BYTE; Initial ver curs pos($WSBL-x)
* Exit 1: hor    BYTE; Ending cursor position
* Exit 2: ver    BYTE; Ending cursor position

If Error: IF  $PUTNOPX (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's).Message is in $MSG externally
          defined buffer. $WCONFIG for screen size.
SPRM Ref: PUTNOPX$       Vol.2 Sec.  7.1
```

$PUTNOTE FUNCTION $PUTNOTE

Workstation, Home-Down Roll, Log/Display Message

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTNOTE  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTNOTE (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTNOTE$      Vol.2 Sec.  7.1
```

$PUTNOTX FUNCTION $PUTNOTX

Workstation, Log/Display Message

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 01
File    : D$UFRWS

Syntax  : $PUTNOTX  (message, &hor, &ver)
Result  : D$CCODE

Entry 1 : message ^ CHAR; Message terminated by $ES
Entry 2 : hor       BYTE; Initial hor curs pos($WSLC)
Entry 3 : ver       BYTE; Initial ver curs pos($WSBL-x)
* Exit 2: hor       BYTE; Ending cursor position
* Exit 3: ver       BYTE; Ending cursor position

If Error: IF  $PUTNOTX (...) && D$CFLAG THEN $ERMSG ();
Remarks : Entry requires $WSIO mode bits in PUTWSMD$
          (usually 0's). $WCONFIG for screen size.
SPRM Ref: PUTNOTX$      Vol.2 Sec.  7.1
```

RASLEND$ FUNCTION RASLEND$

Turn Off RASL Traps in Program Running RASL

Category: DASL External Function
Entered : 83 Jul 19 *Updated* : 84 Jul 25 |*
File : D$INC

Syntax : **RASLEND$** *()*
Result : D$CCODE |*

If Error: **IF RASLEND$ (...) && D$CFLAG THEN $ERMSG ();** |*
Remarks : This function may be called in a program
that initially was running RASL.
There is a separate RASL Document:
The RASL Report: A Symbolic Debugger for DASL

RASLRES$ FUNCTION RASLRES$

Invoke the DASL Debugger

Category: DASL External Function
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$INC

Syntax : **RASLRES$** *()*
Result : D$CCODE |*

If Error: **IF RASLRES$ (...) && D$CFLAG THEN $ERMSG ();** |*
Remarks : The RASLRES$ module can be invoked through
an option on the command line; DASL/CHN;DBUG.
There is a separate RASL Document:
The RASL Report: A Symbolic Debugger for DASL

RECURSIVE FUNCTION RECURSIVE

Specifies Function may be Called Recursively

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **RECURSIVE *func (params) resultType:=statmnt;***
DASL Doc: 24,77,78 March 1982

$RELFAVS FUNCTION $RELFAVS

Release All Locked FAV's

Category: System Call
Entered : 82 Dec 01 *Updated* : 83 Feb 01
File : D$RMSSPEC

Syntax : **$RELFAVS *()***
Result : *D$CCODE*

If Error: **IF $RELFAVS (...) && D$CFLAG THEN $ERMSG ( );**

$RENENV FUNCTION $RENENV

Multi-Resource, Change a Disk File Name

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $RENENV   (openpt)
Result  : D$CCODE

Entry 1 : openpt  ^ $OPENPT; PFDB of open file with     |*
                             new name and HSI           |*
If Error: IF  $RENENV (...) && D$CFLAG THEN $ERMSG ();
             $ERRC.$FUNC = SC$RENENV
             $ERRC.$CODE = Single- and multi-file       |*
                  $ECUMAV,$ECSI001,$ECSI005,$ECSI007,   |*
                  $ECSI010,$ECSI021,$ECSI030,$ECSI031,  |*
                  $ECSI033,$ECSI045                     |*
                           Multi-file resources only    |*
                  $ECSI012,$ECSI013,$ECSI014,$ECSI015,  |*
                  $ECSI016,$ECSI024                     |*
SPRM Ref: $RENENV       Vol.4 Sec.  8.4.6
```

$REOPEN FUNCTION $REOPEN

Multi-Resource, Reopen a File With New Passwords

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $REOPEN   (openpt)
Result  : D$CCODE

Entry 1 : openpt  ^ $OPENPT; of file opened in $OMEXCL  |*
                             mode                       |*

If Error: IF  $REOPEN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$REOPEN
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,      |*
         $ECSI007,$ECSI010,$ECSI012,$ECSI014,$ECSI016,  |*
         $ECSI021,$ECSI025,$ECSI028,$ECSI045,$ECSI053,  |*
SPRM Ref: $REOPEN       Vol.4 Sec.  8.4.2
```

Assign a Value to the Result of a Function

Category: DASL Control Word
Entered : 83 Jul 19 *Updated* :

Syntax : RESULT := ***expression;***

Remarks : This is used when writing the code to define a "function" type variable.
Assigning a value to a Macro function (like all of these RMS System Function interfaces) is done in a DEFINE statement.
If the last expression in the DEFINE statement is NOT an assignment, but an expression (like a variable name or parameter), that value will be assigned to the Macro as a "resultant" value.
DASL Doc: 77,78 February 1983
See Also: DEFINE

$RFIAKS FUNCTION $RFIAKS

Return From Abort Key Seq Interrupt

Category: System Call
Entered : 83 Apr 02 *Updated* : 83 Jul 19
File : D$RMSPROG

Syntax : **$RFIAKS** ***(loc, trc)***
Result : *D$CCODE*

Entry 1 : ***loc*** **^ $INTS;** State table
Entry 2 : ***trc*** **BYTE;** 0 resets trap, or $RFITRC

If Error: **IF $RFIAKS (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$RFI
$ERRC.$CODE = $ECRFI0,$ECRFI1

$RFIDKS FUNCTION $RFIDKS

Return from $TRAPDKS Interrupt

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSPROG

Syntax : **$RFIDKS** ***(loc, trc)***
Result : *D$CCODE*

Entry 1 : ***loc*** **^ $INTS;** State table
Entry 2 : ***trc*** **BYTE;** 0 resets trap, or $RFITRC

If Error: **IF $RFIDKS (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$RFI
$ERRC.$CODE = $ECRFI0,$ECRFI1
See Also: $TRAPSET for general info.;
$TRAPDKS for calling routine
SPRM Ref: $RFIDKS Vol.4 Sec. 4.5.4

$RFIFK FUNCTION $RFIFK

Return from $TRAPFK Interrupt

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSPROG

Syntax : **$RFIFK** ***(loc, trc, noFifoCk)***
Result : *D$CCODE*

Entry 1 : ***loc*** **^ $INTS**; State table
Entry 2 : ***trc*** **BYTE**; 0 resets trap, or $RFITRC
Entry 3 : ***noFifoCk*** **BOOLEAN**; Alternate mode if true

If Error: **IF $RFIFK (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$RFI
$ERRC.$CODE = $ECRFI0,$ECRFI1
Remarks : Nucleus mechanism is such that $RFIFK is ineffective if the FIFO buffer has characters in it, so put a jump to some location in the user program following $RFIFK call.
See Also: $TRAPSET for general info
SPRM Ref: $RFIFK Vol.4 Sec. 4.5.6

$RFIKKS FUNCTION $RFIKKS

Return From $TRAPKKS Interupt

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSPROG

Syntax : **$RFIKKS** ***(loc, trc)***
Result : *D$CCODE*

Entry 1 : ***loc*** **^ $INTS**; State table
Entry 2 : ***trc*** **BYTE**; 0 resets trap, or $RFITRC

If Error: **IF $RFIKKS (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$RFI
$ERRC.$CODE = $ECRFI0,$ECRFI1
See Also: $TRAPSET for general info.
SPRM Ref: $RFIKKS Vol.4 Sec. 4.5.2

Return From $TRAPLKS Interupt

Category: System Call
Entered : 83 Jul 19 *Updated* :
File : D$RMSPROG

Syntax : **$RFILKS** ***(loc, trc)***
Result : *D$CCODE*

Entry 1 : ***loc*** **^ $INTS;** State table
Entry 2 : ***trc*** BYTE; 0 resets trap, or $RFITRC

If Error: **IF $RFILKS (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$RFI
$ERRC.$CODE = $ECRFI0,$ECRFI1
See Also: $TRAPSET for general info.
$TRAPLKS for calling routine

Err-Msg, RETURN RMS MESSAGE

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRERR

Syntax : **$RMSMSG** ***(&length)***
Result : *D$CCODE*

** Exit 1*: ***length*** **BYTE;**

If Error: **IF $RMSMSG (...) && D$CFLAG**
THEN (I/O error);
Remarks : The $ERRC 2 byte variable contains the Function and Error codes obtained from the previous function that was called and detected an error.
$RMSMSG is similar to $MSGC but does not abort by calling $ERROR. This is used to only display errors detected that are not fatal.
See Also: $MSGC for fatal case messages
SPRM Ref: RMSMSG$ Vol.2 Sec. 3.3.5

Load and Run a Program

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25 |*
File    : D$RMSPROG

Syntax  : $RUN  (pfdb, lsn, asIs)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : lsn     UNSIGNED;
Entry 3 : asIs    BOOLEAN;   ?

If Error: IF  $RUN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$RUN
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,    |*
                        $ECSI010                      |*
          Note:
          $RUN does not normally return, it runs a
          new program.  Also can return to Command
          Interpreter with following codes on top of
          stack: $CISMEM,$CISFMT,$CISREAD.
Remarks : $$RUN Should be used to guarantee
          compatibility with interface compatibility
          of CHAIN and LOG utilities.
SPRM Ref: $RUN          Vol.4 Sec.  4.1
```

Workstation, Interface to $RUN System Call

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWS

Syntax : **$$RUN *(pfdb, lsn)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB;**
Entry 2 : ***lsn*** **UNSIGNED;**

If Error: **IF $$RUN (...) && D$CFLAG THEN $ERMSG ();**
Remarks : $$RUN Should be used to guarantee compatibility with interface compatibility of CHAIN and LOG utilities.
See Also: $RUN for error messages
SPRM Ref: RUN$ Vol.2 Sec. 7.22

CmdInt, Read and Scan User Configuration File

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRSCAN

Syntax : **$SCANCFG** ***(pfdb, cfghdr)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB;**
Entry 2 : ***cfghdr*** **^ $CFGHDR;**

If Error:
```
IF  $SCANCFG (...) && D$CFLAG
 THEN {
    IF ($ERRC.$FUNC = SC$SECR |
        $ERRC.$FUNC = SC$SECWAIT)
     THEN {I/O error}
     ELSE {header not found in file};
    };
```

Remarks : used to declare data structures for specifying contents of a user configuration file, scan data and determine results
SPRM Ref: SCANCFG$ Vol.2 Sec. 6.7.4

CmdInt, Scan File Specs According to Table

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Jul 30
File    : D$RMS

Syntax  : $SCANFLS  (fileSpk, num)
Result  : D$CCODE

Entry 1 : fileSpk ^ $FILESPK; Table of file specs
Entry 2 : num       BYTE; Number of specs to scan

If Error: IF  $SCANFLS (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = $UECSFL
          $ERRC.$CODE = $UECSFL0,$UECSFL1,$UECSFL2,
                        $UECSFL3,$UECSFL4
Remarks : Evaluates first by symbolic file names,
          then by positional file names.
SPRM Ref: SCANFLS$      Vol.2 Sec.  6.1.2
```

CmdInt, Scan a File Specification

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 83 Apr 23
File    : D$UFRSCAN

Syntax  : $SCANFS   (&scanpt, sfent)
Result  : D$CCODE
Use     : IF $SCANFS (...) && D$ZFLAG=0
          THEN (invalid terminator);

Entry 1 : scanpt    ^ CHAR;
Entry 2 : sfent     ^ $SFENT;
* Exit 1: scanpt    ^ CHAR;

If Error: none occurs
Remarks : scans a string of operands delimited by
          standard characters (:,/,=) and breaks
          data into 4 right-blank-filled fields.
SPRM Ref: SCANFS$       Vol.2 Sec.  6.1.3
```

$SCANHSI FUNCTION $SCANHSI

Environment, Compress HSI

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRENV

Syntax : **$SCANHSI** ***(source, dest, &send, &dend)***
Result : *D$CCODE*
Assign : ***varRslt***:=**$SCANHSI** ***(source,dest,&send,&dend)***;
IF ***varRslt*** **&& D$CFLAG=0 &** ***varRslt*** **&&**
D$ZFLAG=0 THEN *(non-null HSI scanned);*
IF **varRslt && D$CFLAG=0 &** ***varRslt*** **&&**
D$ZFLAG THEN *(null HSI scanned);*

Entry 1 :	***source***	**^ CHAR;**
Entry 2 :	***dest***	**^ CHAR;**
** Exit 3*:	***send***	**^ CHAR;**
** Exit 4*:	***dend***	**^ CHAR;**

If Error: **IF** ***varRslt*** **&& D$CFLAG**
THEN *(invalid HSI format);*
SPRM Ref: SCANHSI$ Vol.2 Sec. 2.12

$SCANNB FUNCTION $SCANNB

CmdInt, Scan to Next Non-Blank

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRSCAN

Syntax : **$SCANNB** ***()***
Result : *CHAR*

If Error: **No Error Occurs**
SPRM Ref: SCANNB$ Vol.2 Sec. 6.3

$SCANOS FUNCTION $SCANOS

CmdInt, Scan Options Specification

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMS

Syntax : **$SCANOS** *(option)*
Result : *D$CCODE*

Entry 1 : ***option*** **^ $OPTION;** Start of One or more

If Error: **IF $SCANOS (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = $UECSOS
$ERRC.$CODE = $UECSOS0,$UECSOS1,$UECSOS2,
$UECSOS3,$UECSOS4
Remarks : After calling $SCANOS, test fields
$OPTION.$OPTFLG and $OPTION.$OPTVAL
SPRM Ref: SCANOS$ Vol.2 Sec. 6.2.3

$SCANSYM FUNCTION $SCANSYM

CmdInt, Scan a Symbol

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRSCAN

Syntax : **$SCANSYM** *(max, output)*
Result : *D$CCODE*

Entry 1 : ***max*** **BYTE;**
Entry 2 : ***output*** **^ CHAR;**

If Error: **IF $SCANSYM (...) && D$CFLAG**
THEN (symbol was truncated)
Remarks : scans a single symbol from a string and puts
it into output area...terminates on first
delimiter encountered in input string
SPRM Ref: SCANSYM$ Vol.2 Sec. 6.4

CmdInt, Scan Two Word Lists for Matches

Category:	User Function Routine	
Entered :	82 Jul 01	*Updated* : 83 Apr 23
File :	D$UFRSCAN	

Syntax : **$SCANWRD** ***(input, check, &count)***
Result : *D$CCODE*

Entry 1 :	***input***	**^ CHAR;**	potentially duplicated words
Entry 2 :	***check***	**^ CHAR;**	list to check against
** Exit 3*:	***count***	**^ BYTE;**	string of numbers whose position corresponds to the word position in input string and value corresponds to number of occurances found in check string

If Error: **IF $SCANWRD (...) && D$CFLAG THEN**
(duplicate words were found in "input" list);
Remarks : Word lengths limited to 12 characters long...
Word list can be up to 8 words long and are
separated by comma and terminated by $ES
SPRM Ref: SCANWRD$ Vol.2 Sec. 6.5

Numeric, Signed 24-bit Division

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRNUM

Syntax : **$SDIVID3** ***(dvisor,dvidend,&quotnt,&remaindr)***
Result : *D$CCODE*
Use : **IF $SDIVID3** ***(...)*** **&& D$ZFLAG**
THEN *(division by zero attempted);*

Entry 1 :	***dvisor***	**ULONG;**	Divisor
Entry 2 :	***dvidend***	**ULONG;**	Dividend
** Exit 3*:	***quotnt***	**ULONG;**	Quotient
** Exit 4*:	***remaindr***	**ULONG;**	Remainder

If Error: **No Error Occurs**
SPRM Ref: SDIVID3$ Vol.2 Sec. 4.17

Check Operation Status

Category:	System Call
Entered :	82 Jul 01 *Updated* : 84 Jul 25
File :	D$RMSIO

Syntax : **$SECCHK** ***(pfdb, &status)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB;**
** Exit 2*: ***status*** **$SECSTAT;**

If Error: **IF $SECCHK (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$SECCHK
See Also: $SECW for error codes.
$SECWAIT for more efficient use of processor (by other tasks), which does effectively a loop on $SECCHK.

SPRM Ref:	$SECCHK	Vol.4 Sec.	7.9
SPRM Ref:	$SECCHK	Vol.4 Sec.	8.5.3
SPRM Ref:	$SECCHK	Vol.4 Sec.	9.1.2.3
SPRM Ref:	$SECCHK	Vol.4 Sec.	9.2.2.2
SPRM Ref:	$SECCHK	Vol.4 Sec.	9.3.2.5
SPRM Ref:	$SECCHK	Vol.4 Sec.	9.4.2.4
SPRM Ref:	$SECCHK	Vol.4 Sec.	10.3.4

Obtain or Set End Of File Location

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25  |
File    : D$RMSIO

Syntax  : $SECEOF  (mode, pfdb, wByte, &rByte)
Result  : D$CCODE

Entry 1 : mode      BYTE; $EOFGET, $EOFSET, $EOFWRIT
Entry 2 : pfdb    ^ $PFDB;
Entry 3 : wByte     BYTE;
* Exit 4: rByte     BYTE;

If Error: IF  $SECEOF (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SECEOF
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,
                $ECSI007,$ECSI009,$ECSI010,$ECSI012,   |
                $ECSI013,$ECSI015,$ECSI021,$ECSI032,   |
                $ECSI034,$ECSI045                      |
SPRM Ref: $SECEOF       Vol.4 Sec.  7.11
SPRM Ref: $SECEOF       Vol.4 Sec.  8.5.5
SPRM Ref: $SECEOF       Vol.2 Sec. 13.5.3
SPRM Ref: $SECEOF       Vol.2 Sec. 13.5.4
```

Block Read

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25  |*
File    : D$RMSIO

Syntax  : $SECR (pfdb, wait)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : wait    BOOLEAN;

If Error: IF  $SECR (...) && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = SC$SECR
          $ERRC.$CODE = The following error codes are  |*
                        common to $SECR and $SECRO:    |*
                                                       |*
                        $PTFREAD error codes           |*
                $ECUMAV,$ECSI001,$ECSI004,$ECSI005,    |*
                $ECSI006,$ECSI007,$ECSI010,$ECSI016,   |*
                $ECSI017,$ECSI018,$ECSI025,$ECSI029    |*
                $ECSI046                               |*

                        $PTFREWND error codes          |*
                $ECSI001,$ECSI005,$ECSI006,$ECSI007,   |*
                $ECSI010,$ECSI016,$ECSI017,$ECSI018,   |*
                $ECSI025,$ECSI029,                     |*
Remarks : Whenever a $SECR, $SECRO, $SECW, or $SECWO   |*
          is initialized, no other operation may be    |*
          attempted to that FAV except $SECCHK,        |*
          $SECWAIT, or $WAITIO.                        |*
          Status returned immediately will be :        |*
          Operation in Progress IF initialization is   |*
          successful or Operation Complete IF there is |*
          an entry parameter error.                    |*
See Also: $SECW for error codes                        |*
SPRM Ref: $SECR         Vol.4 Sec.  7.7
SPRM Ref: $SECR         Vol.4 Sec.  8.5.1.1
SPRM Ref: $SECR         Vol.4 Sec.  9.1.2.1
SPRM Ref: $SECR         Vol.4 Sec.  9.3.2.2
SPRM Ref: $SECR         Vol.4 Sec.  9.4.2.2
SPRM Ref: $SECR         Vol.4 Sec. 10.3.2
SPRM Ref: $SECR         Vol.2 Sec. 13.5.1
```

Block Read Optimum

```
Category: System Call
Entered : 82 Jul 01                Updated : 84 Jul 25   |*
File    : D$RMSIO

Syntax  : $SECRO  (pfdb, wait)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : wait    BOOLEAN;

If Error: IF  $SECRO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SECRO
Remarks : synchronizes number of sectors to those in a |*
          track boundary                               |*
See Also: $SECR and $SECW for error codes              |*
SPRM Ref: $SECRO         Vol.4 Sec.  7.7
SPRM Ref: $SECRO         Vol.4 Sec.  8.5.1.2
SPRM Ref: $SECRO         Vol.4 Sec.  9.1.2.1
SPRM Ref: $SECRO         Vol.4 Sec.  9.4.2.2
SPRM Ref: $SECRO         Vol.4 Sec. 10.3.2
```

Multi-Resource, Change Disk File Security

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$RMSIO

Syntax : **$SECURE** ***(mode, loc)***
Result : *D$CCODE*

Entry 1 : ***mode*** **BYTE; $SSGET,$SSPUT,$SSGETX,$SSPUTX**
Entry 2 : ***loc*** **^ $SECURETBL;**

If Error: **IF $SECURE (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SECURE
$ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI005,
$ECSI007,$ECSI010,$ECSI012,$ECSI013,
$ECSI014,$ECSI015,$ECSI016,$ECSI021, |*
$ECSI034,$ECSI037,$ECSI042,$ECSI045 |*
Remarks : File must be open in $OMEXCL mode with |*
$ACSECQ set. |*
SPRM Ref: $SECURE Vol.4 Sec. 8.4.7

Block Write

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$RMSIO

Syntax : **$SECW** ***(pfdb, wait)***
Result : *D$CCODE*

Entry 1 : ***pfdb*** **^ $PFDB;**
Entry 2 : ***wait*** **BOOLEAN;**

If Error: **IF $SECW (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SECW
$ERRC.$CODE = ...see next page. |*

.... *continued*

$SECR, $SECRO, $SECW, $SECWO, $SECWAIT, |*
and **$SECCHK** share the following error codes:|*

<u>All resources</u>
    $ECUMAV,$ECSI001,$ECSI002,$ECSI003, |*
    $ECSI004,$ECSI005,$ECSI007,$ECSI010, |*
    $ECSI017,$ECSI039 |*

<u>Disk resources</u>
    $ECSI006,$ECSI008,$ECSI009,$ECSI011, |*
    $ECSI012,$ECSI013,$ECSI014,$ECSI015, |*
    $ECSI016,$ECSI018,$ECSI026,$ECSI045, |*
    $ECSI046,$ECSI058 |*

<u>Pipe resources</u>
    $ECSI016,$ECSI019,$ECSI045,$ECSI051, |*
    $ECSI052 |*

<u>Printer resources</u>
    $ECSI045,$ECSI062 |*

<u>Tape resources</u>
    $ECSI006,$ECSI009,$ECSI016,$ECSI018, |*
    $ECSI019,$ECSI046 |*

<u>Card reader resources</u>
    $ECSI008,$ECSI016,$ECSI047,$ECSI048 |*

<u>Comm resources</u>
    $ECSI044 |*

<u>Receiving a RIM message</u>
    $ECUMAV,$ECSI001,$ECSI002,$ECSI003, |*
    $ECSI004,$ECSI005,$ECSI007,$ECSI010, |*
    $ECSI017,$ECSI039,$ECSI045 |*

The following error codes are common |*
ONLY to **$SECW** and **$SECWO**: |*

<u>$DC150 error codes</u>
    $ECUMAV,$ECSI001,$ECSI004,$ECSI005, |*
    $ECSI006,$ECSI007,$ECSI009,$ECSI010, |*
    $ECSI016,$ECSI017,$ECSI018,$ECSI025, |*
    $ECSI029,$ECSI046 |*

SPRM Ref:	$SECW	Vol.4 Sec. 7.8
SPRM Ref:	$SECW	Vol.4 Sec. 8.5.2.1
SPRM Ref:	$SECW	Vol.4 Sec. 9.1.2.2
SPRM Ref:	$SECW	Vol.4 Sec. 9.2.2.1
SPRM Ref:	$SECW	Vol.4 Sec. 9.3.2.3
SPRM Ref:	$SECW	Vol.4 Sec. 9.4.2.3
SPRM Ref:	$SECW	Vol.4 Sec. 10.3.3
SPRM Ref:	$SECW	Vol.2 Sec. 13.5.2

Wait for Operation Complete

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |
File    : D$RMSIO

Syntax  : $SECWAIT  (pfdb, &status)
Result  : D$CCODE

Entry 1 : pfdb    ^ $PFDB;
* Exit 2: status    $SECSTAT;

If Error: IF  $SECWAIT (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SECWAIT
See Also: $SECW for error codes                           |
SPRM Ref: $SECWAIT      Vol.4 Sec.  7.10
SPRM Ref: $SECWAIT      Vol.4 Sec.  8.5.4
SPRM Ref: $SECWAIT      Vol.4 Sec.  9.1.2.3
SPRM Ref: $SECWAIT      Vol.4 Sec.  9.2.2.3
SPRM Ref: $SECWAIT      Vol.4 Sec.  9.3.2.4
SPRM Ref: $SECWAIT      Vol.4 Sec.  9.4.2.5
SPRM Ref: $SECWAIT      Vol.4 Sec. 10.3.5
```

$SECWO FUNCTION $SECWO

Block Write Optimum

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Aug 08
File    : D$RMSIO

Syntax  : $SECWO  (pfdb, wait)
Result  : D$CCODE

Entry 1 : pfdb  ^ $PFDB;
Entry 2 : wait    BOOLEAN;

If Error: IF  $SECWO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SECWO
Remarks : synchronizes number of sectors to those in    |*
          track boundary
See Also: $SECW for error codes                         |*

SPRM Ref: $SECWO        Vol.4 Sec.  7.8
```

SET FUNCTION SET

Define Variable Type BYTE and Define Powers of 2

```
Category: DASL Include Macro
Entered : 82 Jul 01                 Updated : 83 Jul 10
File    : D$INC

Syntax  : SET  (arg1, arg2,....arg8)
Result  : BYTE
Use     : TYPDEF xvar SET (arg1, arg2,....arg8); or
                 xvar SET (arg1, arg2,....arg8);
Remarks : TYPDEF xvar SET (arg1, arg2,....arg8);
          Equivalent to:
          TYPDEF xvar BYTE;
          DEFINE(arg1,1)
          DEFINE(arg2,2)
          DEFINE(arg3,4)
          ...
          DEFINE(arg8,128)
See Also: SETV for Ascending powers of two
          SETW for unsigned variables
```

$SETABTF FUNCTION $SETABTF

Workstation, Set CHAIN Abort Flag

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |
File : D$UFRWS

Syntax : **$SETABTF** ***(flag)***
Result : *none*

Entry 1 : ***flag*** BYTE; $PCRAFGA in PCR |

If Error: **No Error Occurs**
See Also: CHAIN Language in RMS User's Guide
SPRM Ref: SETABTF$ Vol.2 Sec. 7.17

$SETIME FUNCTION $SETIME

Set The Current System Time

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 01 |
File : D$RMSGEN

Syntax : **$SETIME** ***(src, utc)***
Result : *D$CCODE*

Entry 1 : ***src*** **^ $SETTIMEP;**
Entry 2 : ***utc*** **BOOLEAN;** Use Alternate SC if TRUE

If Error: **IF $SETIME (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$SETIME
$ERRC.$CODE = $ECUMAV,$ECSTM1,$ECSTM2,
$ECSTM3 |
Remarks : 9 byte field used to compensate for accuracy
ageing, etc. of oscillator for system clock
as well as to set clock
When you use the alternate System Call, you |
must use the syntax: |
$SETIME (<^$SETIMEP> *dest*, **TRUE)** |
where: *dest* is a **^$SYSTINFO.** |
SPRM Ref: $SETIME Vol.4 Sec. 3.2

$SETMAX FUNCTION $SETMAX

Set Maximum Memory Requirement

```
Category: System Call
Entered : 83 May 03                Updated : 84 Jul 25   |*
File    : D$RMSMEM

Syntax  : $SETMAX  (max, &old)
Result  : D$CCODE

Entry 1 : max     BYTE; Maximum 4k sectors to reserve
* Exit 2: old     BYTE; Previous number reserved

If Error: IF  $SETMAX (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SETMAX
          $ERRC.$CODE = $ECSMAX0,$ECSMAX1                |*
```

$SETMIN FUNCTION $SETMIN

Set Minimum Memory Requirement

```
Category: System Call
Entered : 82 Jul 01                Updated : 84 Jul 25   |*
File    : D$RMSMEM

Syntax  : $SETMIN  (nSec, &old)
Result  : D$CCODE

Entry 1 : nSec    BYTE; 4k sectors to be reserved
* Exit 2: old     BYTE; Previous number reserved

If Error: IF  $SETMIN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SETMIN
          $ERRC.$CODE = $ECSMIN0,$ECSMIN1                |*
SPRM Ref: $SETMIN          Vol.4 Sec.  5.2.6
```

$SETPRI FUNCTION $SETPRI

Set User Task Priority Level

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 01
File : D$RMSPROG

Syntax : **$SETPRI** ***(pri, &prevPri)***
Result : *D$CCODE*

Entry 1 : ***pri*** BYTE; 0 to $PRIMAX ($PRINORM)
* *Exit 2*: ***prevPri*** BYTE; 0 is highest level

If Error: **No Error Occurs**
SPRM Ref: $SETPRI Vol.4 Sec. 4.6

$SETSQL FUNCTION $SETSQL

Set Independent Task Security Level

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |
File : D$RMSGEN

Syntax : **$SETSQL** ***(sql, &old)***
Result : *D$CCODE*

Entry 1 : ***sql*** BYTE; 0 to $SQLMAX
* *Exit 2*: ***old*** BYTE; see also: $SQLCHEK, $SQLREPR

If Error: **No Error Occurs**
REMARKS : can be used to determine or set a task's
security level, but cannot raise it above |
user's task level. |
SPRM Ref: $SETSQL Vol.4 Sec. 3.3

Define Ascending Powers of 2 from Initial Value

Category: DASL Include Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$INC

Syntax : SETV ***(initialValue, arg1, arg2,....arg8)***
Result : *Defines values of argument identifiers*

Remarks : SETV is similar to SET except that
a) Defined values begin at the initialValue and may be as large as (2 to the 15th).
b) SETV does not define a variable type.
See Also: SET for powers of two and variable type BYTE
SETW for unsigned variables
DASL Doc: 90 March 1982

Define Variable Type UNSIGNED; Define Powers of 2

Category: DASL Include Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$INC

Syntax : SETW ***(arg1, arg2, ...arg8)***
Result : *UNSIGNED*
Use : TYPDEF ***xvar*** SETW ***(arg1, arg2, ...arg8);***
or ***xvar*** SETW ***(arg1, arg2, ...arg8);***

Remarks : SETW is like SET except result is UNSIGNED
See Also: SET for powers of two and variable type BYTE
SETV for Ascending powers of two
DASL Doc: 90 March 1982

Force User Signon

Category: System Call
Entered : 83 Apr 02 *Updated* :
File : D$RMSPROG

Syntax : **$SIGNON** *()*
Result : *D$CCODE*

If Error: **IF $SIGNON (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SIGNON
$ERRC.$CODE = ?

Start an Independent Task

```
Category: System Call
Entered : 82 Jul 01                Updated : 84 Jul 25   |*
File    : D$RMSTASK

Syntax  : $SINDEP  (name, pfdb, pcrs, &secret, &id)
Result  : D$CCODE

Entry 1 : name ^ $NAMET; Symbolic name of task,
                         Must be unique at node
Entry 2 : pfdb ^ $PFDB; PFDB of file to be loaded
Entry 3 : pcrs   BYTE; MSN to be used as task PCR
                       (Initialized by $NEWPCR )
* Exit 4: secret UNSIGNED; The task's unique secret
                         number; to use with $TASKCTL
* Exit 5: id     BYTE; If "secret" is not zero ? then
                       this is tasks identification

If Error: IF  $SINDEP (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SINDEP
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECTSK2,        |*
                $ECTSK4,$ECSI005,$ECTSK6,$ECTSK7,        |*
                $ECTSK8,$ECTSK9,$ECSI010,$ECTSK11,       |*
                $ECTSK12,$ECTSK13,$ECTSK14,$ECTSK15,     |*
                $ECTSK16,$ECTSK17,$ECTSK18,$ECSI030,     |*
                $ECSI031                                 |*
Remarks : Can return to CMD INT with codes on stack:
          Error Codes: $CISFMT,$CISREAD,$CISMEM,
          $CISADR,$CISACC,$CISDUAL,$CISHAR,$CISYSTB,
          $CISVABT
          Logoff Message Numbers:$CILOG0 thru $CILOG9
SPRM Ref: $SINDEP       Vol.4 Sec.  6.4.1
```

SIZEOF FUNCTION SIZEOF

Operator Which Gives Size in Bytes of Argument

Category: DASL Reserved Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **SIZEOF** ***[expression] or [< type >]***
Result : *UNSIGNED constant*

DASL Doc: 68 March 1982

$SLOCAL FUNCTION $SLOCAL

Start Local Task

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |
File : D$RMSTASK

Syntax : **$SLOCAL** ***(name, sadr)***
Result : *D$CCODE*

Entry 1 : ***name*** **^ $NAMET;**
Entry 2 : ***sadr*** **$STARTADR;**

```
If Error: IF  $SLOCAL (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SLOCAL
          $ERRC.$CODE = $ECUMAV,$ECTSK3,$ECTSK4,
               $ECSI030,$ECSI031                  |
```
SPRM Ref: $SLOCAL Vol.4 Sec. 6.4.2

$SMLPLY3 FUNCTION $SMLPLY3

Numeric, Signed 24-bit Multiplication

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 23
File : D$UFRNUM

Syntax : **$SMLPLY3** ***(val1, val2, &result)***
Result : *D$CCODE*

Entry 1 :	***val1***	**ULONG;** Signed Multiplicand
Entry 2 :	***val2***	**ULONG;** Signed Multiplier
** Exit 3*:	***result***	**ULONG;** Signed Product

If Error: **IF $SMLPLY3 (...) && D$CFLAG**
THEN *(overflow on multiplication);*
SPRM Ref: SMLPLY3$ Vol.2 Sec. 4.16

STATIC FUNCTION STATIC

Prevents Re-allocation of Var. in Recur. Function

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : ***VAR STATIC*** **name type ;**

Remarks : The variable may also be initialized in the declaration statement. However, any static variable will be initialized only the first time the statement is executed. If the statement is inside a function definition, the variable will retain its previous value when the function is called after the first time. (the variable may be ASSIGNED new values like any other variable.)

DASL Doc: 25,75,76 March 1982

Stop All Data Movement

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Aug 08   |*
File    : D$RMSIO

Syntax  : $STOPIO   (mode, pfdb)
Result  : D$CCODE

Entry 1 : mode     BYTE; $STOPALL, $STOPONE
Entry 2 : pfdb   ^ $PFDB; required for only $STOPONE      |*

If Error: IF  $STOPIO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$STOPIO
          $ERRC.$CODE = $ECUMAV,$ECSI001
Remarks : Forces all I/O operations to complete ASAP.   |*
          $SECCHK or $SECWAIT can be used to see         |*
          operation status ($SECSTOP will be set at      |*
          completion).                                   |*
SPRM Ref: $STOPIO       Vol.4 Sec.  7.13
SPRM Ref: $STOPIO       Vol.4 Sec.  8.5.7
SPRM Ref: $STOPIO       Vol.4 Sec.  9.1.2.5
SPRM Ref: $STOPIO       Vol.4 Sec.  9.2.2.5
SPRM Ref: $STOPIO       Vol.4 Sec.  9.4.2.7
SPRM Ref: $STOPIO       Vol.4 Sec. 10.3.7
```

STRUCT FUNCTION STRUCT

Named Member Consisting of Several Named Members

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : ***x*** **STRUCT** ***{var type; var type;[..var type;]};***

Remarks : A structure is any variable made up of more than one variable.
DASL Doc: 16,59,60 March 1982

SUBSTR FUNCTION SUBSTR

Select Part of String, Begin at Start for Length

Category: DASL Compiler Macro
Entered : 82 Jul 01 *Updated* : 83 Jul 10

Syntax : **SUBSTR** ***(string, startNumber, lengthNumber)***
Result : *part of string, (0 is first character)*

See Also: DEFINE for string definition
DASL Doc: 29,33,81-82 March 1982

SYSTEM FUNCTION SYSTEM

Reserved for Future Code Generators

Category: DASL Reserved Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **None: *Reserved Word***

$TAPEREWIND FUNCTION $TAPEREWIND

Rewind Tape

Category: User Function Routine
Entered : 83 Apr 02 *Updated* : 83 May 03
File : D$UFRWFIO

Syntax : **$TAPEREWIND *(fcb)***
Result : *D$CCODE*

Entry 1 : ***fcb* ^ $WFCB;**

If Error: IF $TAPEREWIND (...) && D$CFLAG THEN $ERMSG ();

$TAPEUNLOAD FUNCTION $TAPEUNLOAD

Rewind Tape and Unload

```
Category: User Function Routine
Entered : 83 Apr 02                  Updated : 83 May 03
File    : D$UFRWFIO

Syntax  : $TAPEUNLOAD  (fcb)
Result  : D$CCODE

Entry 1 : fcb  ^ $WFCB;

If Error: IF $TAPEUNLOAD (...) && D$CFLAG THEN $ERMSG ( );
```

$TASKCTL FUNCTION $TASKCTL

Exert Control Over an Independent Task

```
Category: System Call
Entered : 83 Apr 02                  Updated : 84 Jul 25   |*
File    : D$RMSTASK

Syntax  : $TASKCTL  (mode, taskId, secretNm)
Result  : D$CCODE

Entry 1 : mode      BYTE; $TCLOKS,$TCAKS,$TCDKS,
                          $TCKKS,$TCFK
Entry 2 : taskId    BYTE; from $SINDEP
Entry 3 : secretNm  UNSIGNED; from $SINDEP

If Error: IF  $TASKCTL (...) && D$CFLAG THEN $ERMSG ( );
          $ERRC.$FUNC = SC$TASKCTL
          $ERRC.$CODE = $ECTCL0,$ECTCL1,$ECTCL2,       |*
                $ECTCL3                                |*
```

THEN FUNCTION THEN

Part of IF THEN ELSE Execution Control

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 83 Jul 10

See Also: IF for usage syntax
DASL Doc: 73 March 1982

$TIMER FUNCTION $TIMER

Reset System Timer

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TIMER  (time, pfdb)
Result  : D$CCODE

Entry 1 : time   BYTE;
Entry 2 : pfdb ^ $PFDB;

If Error: IF $TIMER (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$TIMER
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECSI007,
                        $ECSI039,$ECSI041
SPRM Ref: $TIMER        Vol.4 Sec.  4.7
```

$TRAPAKS FUNCTION $TRAPAKS

Trap Abort Key Sequence

```
Category: System Call
Entered : 83 Apr 02                  Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TRAPAKS  (loc, &prev)
Result  : D$CCODE

Entry 1 : loc   ^ $INTS; State Table
* Exit 2: prev  ^ $INTS;

If Error: IF  $TRAPAKS (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$TRAPSET                         |*
          $ERRC.$CODE = $ECTRAPO                           |*
See Also: $TRAPSET for general info. and $RFIAKS for      |*
          return routine                                   |*
```

$TRAPDKS FUNCTION $TRAPDKS

Trap DISPLAY-CANCEL-DISPLAY Key Sequence

```
Category: System Call
Entered : 82 Jul 01                  Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TRAPDKS  (loc, &prev)
Result  : D$CCODE

Entry 1 : loc     ^ $INTS; State table
* Exit 2: prev    ^ $INTS;

If Error: IF  $TRAPDKS (...) && D$CFLAG THEN $ERMSG ();|*
          $ERRC.$FUNC = SC$TRAPSET                         |*
          $ERRC.$CODE = $ECTRAPO                           |*
See Also: $TRAPSET for general info. and $RFIDKS for      |*
          return routine                                   |*
SPRM Ref: $TRAPDKS      Vol.4 Sec.  4.5.3
```

$TRAPFK FUNCTION $TRAPFK

Trap Function Key Strokes

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TRAPFK   (loc, &prev)
Result  : D$CCODE

Entry 1 : loc     ^ $INTS; State table
* Exit 2: prev    ^ $INTS;

If Error: IF  $TRAPFK (...) && D$CFLAG THEN $ERMSG ( );  |*
          $ERRC.$FUNC = SC$TRAPSET                        |*
          $ERRC.$CODE = $ECTRAPO                          |*
See Also: $TRAPSET for general info. and $RFIFK for       |*
          return routine                                  |*
SPRM Ref: $TRAPFK        Vol.4 Sec.  4.5.5
```

$TRAPKKS FUNCTION $TRAPKKS

Trap Keyboard Key Sequence

```
Category: System Call
Entered : 82 Jul 01                 Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TRAPKKS  (loc, &prev)
Result  : D$CCODE

Entry 1 : loc     ^ $INTS; State table
* Exit 2: prev    ^ $INTS;

If Error: IF  $TRAPAKS (...) && D$CFLAG THEN $ERMSG ( );?
          $ERRC.$FUNC = SC$TRAPSET                        |*
          $ERRC.$CODE = $ECTRAPO                          |*
See Also: $TRAPSET for general info. and $RFIKKS for      |*
          return routine                                  |*
SPRM Ref: $TRAPKKS       Vol.4 Sec.  4.5.1
```

Trap LOG-OFF Key Sequence

```
Category: System Call
Entered : 83 Jul 19                Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $TRAPLKS  (loc, &prev)
Result  : D$CCODE

Entry 1 : loc     ^ $INTS; State table
* Exit 2: prev    ^ $INTS;

If Error: IF  $TRAPLKS (...) && D$CFLAG THEN $ERMSG ();|*
          $ERRC.$FUNC = SC$TRAPSET                     |*
          $ERRC.$CODE = $ECTRAP0                       |*
See Also: $TRAPSET for general info. and $RFILKS for   |*
          return routine                               |*
```

Trap Setting System Call, Used Indirectly

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |*
File : D$RMSPROG

Syntax : **DASL *FUNCTION DEFINITION MACRO***

Remarks : Used to define the second parameter passed to SYSTEM CALL (40) by several TRAP SETTING DASL Function Macros.

NOTE:
If more than one trap is set concurrently, each must have its own private $INTS state table (see $RFI...). |*

FAST CODE: Trap routine code must be declared FAST "code."

RESULT: This prevents a stack problem with $RFI. |*

See Also: $TRAPDKS $TRAPFK $TRAPKKS $TRAPUMV $TRAPLKS for specific traps, and $RFIDKS $RFIFK $RFIKKS $RFILKS for returns.
SPRM Ref: (Traps) Vol.4 Sec. 4.5

$TRAPUMV FUNCTION $TRAPUMV

Trap User Mode Violations, Used Indirectly

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSPROG

Syntax : **$TRAPUMV** ***(loc, &prev)***
Result : *D$CCODE*

Entry 1 : ***loc*** **$STARTADR;** Trap routine($NOADR clears)
** Exit 2*: ***prev*** **$STARTADR;**

If Error: **No Error Occurs**
Remarks : The trap location is called if a user task has one of following: $UTEWRIT,$UTEACCS, $UTEINST,$UTEUNDF,$UTEHALT. Calling $GLUTEN in the trap routine will get the last user task error number that caused the trap.
See Also: $GLUTEN for info on task error number trapped
SPRM Ref: $TRAPUMV Vol.4 Sec. 4.5.7

$TXBKSP FUNCTION $TXBKSP

Backspace a logical record

Category: User Function Routine
Entered : 83 Apr 02 *Updated* : 84 Jul 25 |*
File : D$UFRWFIO

Syntax : **$TXBKSP** ***(fcb)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**

If Error: **IF $TXBKSP (...) && D$CFLAG THEN $ERMSG ();**

See also: $SEC... system call error codes |*

Terminate Processing for a Text-File

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 82 Dec 01
File    : D$UFRWFIO

Syntax  : $TXCLOSE  (fcb)
Result  : D$CCODE

Entry 1 : fcb  ^ $WFCB;

If Error: IF  $TXCLOSE (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$SECW,SC$SECWAIT or SC$SECEOF
          $ERRC.$CODE = see $SECW,$SECWAIT,$SECEOF
SPRM Ref: TXCLOSE$      Vol.2 Sec. 13.3.3
```

Delete a Logical Text Record

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 82 Dec 01
File    : D$UFRWFIO

Syntax  : $TXDEL    (fcb)
Result  : D$CCODE
Assign  : varRslt := $TXDEL    (fcb);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN ( operation complete );
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN ( attempt to delete EOF record );

Entry 1 : fcb  ^ $WFCB;

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
See also: $SEC... system call error codes                   |*
SPRM Ref: TXDEL$        Vol.2 Sec. 13.3.10
```

$TXOPEN FUNCTION $TXOPEN

Prepare an Opened Text-File for Access

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$TXOPEN** ***(fcb, openpt)***
Result : *D$CCODE*

Entry 1 : ***fcb*** ^ **$WFCB;**
Entry 2 : ***openpt*** ^ **$OPENPT;** of $OPEN'ed file

If Error: **IF $TXOPEN (...) && D$CFLAG THEN $ERMSG;** |*
SPRM Ref: TXOPEN$ Vol.2 Sec. 13.3.2

$TXOPENP FUNCTION $TXOPENP

Open Using Specified Physical I/O Routine

Category: User Function Routine
Entered : 83 Apr 02 *Updated* : 84 Jul 25 |*
File : D$UFRWFIO

Syntax : **$TXOPENP** ***(fcb, openpt)***
Result : *D$CCODE*

Entry 1 : ***fcb*** ^ **$WFCB;**
Entry 2 : ***openpt*** ^ **$OPENPT;** of $OPEN'ed file

If Error: **IF $TXOPENP (...) && D$CFLAG THEN $ERMSG ( );**
See Also: $OPENENV for error codes. |*
|*

Position to Text-File EOF

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$TXPOSEF** ***(fcb)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**

If Error: **No Error Occurs**
SPRM Ref: TXPOSEF$ Vol.2 Sec. 13.3.5

Position Text-File to File Pointer

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$TXPOSIT** ***(fcb, fp)***
Result : *D$CCODE*
Assign : ***varRslt*** := **$TXPOSIT** ***(fcb, fp);***
IF ***varRslt*** **&&** D$CFLAG=0 **&** ***varRslt*** **&&** D$ZFLAG
THEN *(pointer given is current EOF);*

Entry 1 : ***fcb*** **^ $WFCB;**
Entry 2 : ***fp*** **^ $FILEPTR;**

If Error: IF ***varRslt*** **&&** D$CFLAG
THEN *(pointer given is beyond EOF);*
SPRM Ref: TXPOSIT$ Vol.2 Sec. 13.3.6

$TXPREP FUNCTION $TXPREP

Prepare a New Text-File for Access

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 84 Aug 08   |*
File    : D$UFRWFIO

Syntax  : $TXPREP   (fcb, openpt)
Result  : D$CCODE

Entry 1 : fcb    ^ $WFCB;
Entry 2 : openpt ^ $OPENPT; of $OPEN'ed file

If Error: IF $TXPREP (...) && D$CFLAG THEN $ERMSG ();   |*

See also: $SEC... & $OPENENV system call error codes    |*
SPRM Ref: TXPREP$       Vol.2 Sec. 13.3.1
```

$TXPREPP FUNCTION $TXPREPP

Prepare Using Specified Physical I/O Routine

```
Category: User Function Routine
Entered : 83 Apr 02                Updated : 84 Aug 08   |*
File    : D$UFRWFIO

Syntax  : $TXPREPP  (fcb, openpt)
Result  : D$CCODE

Entry 1 : fcb    ^ $WFCB;
Entry 2 : openpt ^ $OPENPT; of $OPEN'ed file

If Error: IF  $TXPREPP (...) && D$CFLAG THEN $ERMSG ();
See also: $SEC... & $OPENENV system call error codes    |*
```

Read a Logical Text-File Record

```
Category: User Function Routine
Entered : 82 Jul 01               Updated : 84 Jul 25  |*
File    : D$UFRWFIO

Syntax  : $TXREAD  (fcb, rec, size, &end)
Result  : D$CCODE
Assign  : varRslt := $TXREAD  (fcb, rec, size, &end);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN ( record delivered );
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (EOF encountered);

Entry 1 : fcb   ^ $WFCB;
Entry 2 : rec   ^ CHAR; record area for read
Entry 3 : size    UNSIGNED; Maximum record size allowed
                           (with $LEOR)
* Exit 4: end   ^ CHAR; Last character stored,
                        usually $LEOR or $LEOF

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
See Also: $SEC... for error codes                    |*
SPRM Ref: TXREAD$      Vol.2 Sec. 13.3.7
```

$TXUPDATE FUNCTION $TXUPDATE

Update a Logical Text Record

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$UFRWFIO

Syntax : **$TXUPDATE** ***(fcb, rec)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**
Entry 2 : ***rec*** **^ CHAR;** record area terminated by $LEOR

If Error: **IF $TXUPDATE (...) && D$CFLAG THEN $ERMSG ();**
See Also: $SEC... for error codes |*
Remarks : Will not update if disk record has
$LDEL characters.
SPRM Ref: TXUPDAT$ Vol.2 Sec. 13.3.9

$TXWEOF FUNCTION $TXWEOF

Write EOF at Current Text-File Position

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$TXWEOF** ***(fcb)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**

If Error: **IF $TXWEOF (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SECR,SC$SECW or SC$SECWAIT
$ERRC.$CODE = (see $SECR,$SECW,$SECWAIT)
SPRM Ref: TXWEOF$ Vol.2 Sec. 13.3.4

$TXWRITB FUNCTION $TXWRITB

Write Text-file Record with specified length

```
Category: User Function Routine                                |*
Entered : 84 Jul 15               Updated :                    |*
File    : D$UFRWFIO                                            |*

Syntax  : $TXWRITEB  (fcb, rec, recSize)                       |*
Result  : D$CCODE                                              |*

Entry 1 : fcb   ^ $WFCB;                                       |*
Entry 2 : rec   ^ CHAR; record area terminated by $LEOR        |*
Entry 3 : recSize  UNSIGNED;                                   |*

If Error: IF $TXWRITB (...) && D$CFLAG THEN $ERMSG (); |*
See Also: $SEC... for error codes                              |*
```

$TXWRITE FUNCTION $TXWRITE

Write a Logical Text-File Record

```
Category: User Function Routine
Entered : 82 Jul 01               Updated : 84 Jul 25   |*
File    : D$UFRWFIO

Syntax  : $TXWRITE  (fcb, rec)
Result  : D$CCODE

Entry 1 : fcb   ^ $WFCB;
Entry 2 : rec   ^ CHAR; record area terminated by $LEOR

If Error: IF  $TXWRITE (...) && D$CFLAG THEN $ERMSG ();
See Also: $SEC... for error codes                        |*
Remarks : Will write only if disk record has $LDEL's.
SPRM Ref: TXWRITE$      Vol.2 Sec. 13.3.8
```

TYPDEF FUNCTION TYPDEF

Give a Type a Name that may be Used as a Type

```
Category: DASL Declaration Word
Entered : 82 Jul 01               Updated : 82 Dec 01

Syntax  : TYPDEF  name type;
DASL Doc: 17,58,75                     March 1982
```

$UCSCHK FUNCTION $UCSCHK

Check a User Created Semaphore

```
Category: System Call
Entered : 82 Jul 01                  Updated : 82 Dec 01
File    : D$RMSTASK

Syntax  : $UCSCHK   (ucs)
Result  : D$CCODE

Entry 1 : ucs  UNSIGNED;

If Error: IF  $UCSCHK (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$UCSCHK
          $ERRC.$CODE = $ECUCS0,$ECUCS1
SPRM Ref: $UCSCHK       Vol.4 Sec.  6.5.4
```

$UCSDEL FUNCTION $UCSDEL

Delete a User Created Semaphore

```
Category: System Call
Entered : 82 Jul 01                  Updated : 84 Jul 01
File    : D$RMSTASK

Syntax  : $UCSDEL   (ucs)
Result  : D$CCODE

Entry 1 : ucs  UNSIGNED;

If Error: IF  $UCSDEL (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$UCSDEL                           |*
          $ERRC.$CODE = $ECUCS0,$ECUCS1,$ECUCS2             |*
```

$UCSGEN   FUNCTION $UCSGEN

Generate a User Created Semaphore

```
Category: System Call
Entered : 82 Jul 01                    Updated : 83 Apr 02
File    : D$RMSTASK

Syntax  : $UCSGEN  (&ucs)
Result  : D$CCODE

* Exit 1: ucs    UNSIGNED;

If Error: IF  $UCSGEN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$UCSGEN
          $ERRC.$CODE = $ECUCSG0
SPRM Ref: $UCSGEN       Vol.4 Sec.  6.5.1
```

$UCSSIG   FUNCTION $UCSSIG

Signal a User Created Semaphore

```
Category: System Call
Entered : 82 Jul 01                    Updated : 82 Dec 01
File    : D$RMSTASK

Syntax  : $UCSSIG  (ucs)
Result  : D$CCODE

Entry 1 : ucs  UNSIGNED;

If Error: IF  $UCSSIG (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$UCSSIG
          $ERRC.$CODE = $ECUCS0,$ECUCS1
SPRM Ref: $UCSSIG       Vol.4 Sec.  6.5.2
```

$UCSWAIT FUNCTION $UCSWAIT

Wait on a User Created Semaphore

```
Category: System Call
Entered : 82 Jul 01              Updated : 82 Dec 01
File    : D$RMSTASK

Syntax  : $UCSWAIT  (ucs)
Result  : D$CCODE

Entry 1 : ucs  UNSIGNED;

If Error: IF  $UCSWAIT (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$UCSWAIT
          $ERRC.$CODE = $ECUCS0,$ECUCS1
SPRM Ref: $UCSWAIT      Vol.4 Sec.  6.5.3
```

$UERMSG FUNCTION $UERMSG

Store User Error Message on Command Stack

```
Category: User Function Routine                                |*
Entered : 84 Jul 15              Updated :                     |*
File    : D$UFRERR                                             |*

Syntax  : $UERMSG  (message)                                   |*
Result  : Does not return                                      |*

Entry 1 : message ^ CHAR; User's error message will be |*
                          truncated if longer than     |*
                          error message area available |*

If Error: No error occurs                                      |*
Remarks : The Command Interpreter will display the             |*
          message.                                             |*
          $PCREFTS will be set to indicate presence of |*
          error message.                                       |*
          $PCRCLEL will point to 'message'                     |*
```

UNION FUNCTION UNION

Named Member Contains Different Possible Members

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : ***x* UNION *{ var type; var type;[..var type;]};***

Remarks : Allocates memory large enough to hold longest member group.
DASL Doc: 16,59,60 March 1982

$UNLKRIM FUNCTION $UNLKRIM

Release RIM from Pipe ? (on hold)

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRSYS

Syntax : **$UNLKRIM *(pfdb)***
Result : *none*

Entry 1 : ***pfdb*** **^ $PFDB;** Same as used by $LOCKRIM

If Error: **?**
See Also: $LOCKRIM for Pipe opening, RIM lockout
SPRM Ref: $LOCKRIM Vol.2 Sec. 5.6.2

$UNPAKPW FUNCTION $UNPAKPW

Environment, Unpack password into ASCII string

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$UFRENV

Syntax  : $UNPAKPW  (comp, uncomp)
Result  : D$CCODE
Assign  : varRslt := $UNPAKPW  (comp, uncomp);
          IF varRslt && D$CFLAG=0 THEN (valid password);

Entry 1 : comp   ^ $PACKPW;
Entry 2 : uncomp ^ $UNPACKPW;

If Error: IF varRslt && D$CFLAG THEN (null password);
SPRM Ref: UNPAKPW$      Vol.2 Sec.  2.11
```

$USRABN FUNCTION $USRABN

User Abend Facility

```
Category: System Call
Entered : 83 Apr 02                 Updated : 84 Jul 25   |*
File    : D$RMSPROG

Syntax  : $USRABN  (mode, fav, name, &prev)
Result  : D$CCODE

Entry 1 : mode   BYTE; $UABSET,$UABSETO,$UABCLR
Entry 2 : fav    UNSIGNED;
Entry 3 : name ^ $NAMET;
* Exit 4: prev   UNSIGNED;

If Error: IF  $USRABN (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$USRABN
          $ERRC.$CODE = $ECUMAV,$ECSI001,$ECUAB2,        |*
                $ECUAB3,$ECUAB4,$ECUAB5,$ECUAB6,         |*
                $ECSI007,$ECSI010                        |*
```

Indicates Local Variable Definitions Follow

Category: DASL Declaration Word
Entered : 82 Jul 01 *Updated* : 83 Jul 19

Syntax : **VAR** ***varname [...,varname] type;***
... [varname type;]
or
VAR STATIC ***varname type [:= initializer];***

Remarks : Local variables that are initialized in a declaration statement, must be declared STATIC. But, variables declared STATIC do not necessarily need to be intialized.

DASL Doc: 3 March 1982

$VGETBUF FUNCTION $VGETBUF

Obtain Buffer Group from Virtual Pool

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Apr 02
File : D$UFRWFIO

Syntax : $VGETBUF ***(&psk, &startPage, &size)***
Result : *D$CCODE*

* *Exit 1*:	***psk***	BYTE;	
* *Exit 2*:	***startPage***	BYTE;	starting page of PSK
* *Exit 3*:	***size***	BYTE;	buffer group size in pages

If Error: **IF $VGETBUF (...) && D$CFLAG**
THEN (no buffers available)
SPRM Ref: VGETBUF$ Vol.2 Sec. 13.4.7

$VINIT FUNCTION $VINIT

Initialize Virtual I/O Management

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 30
File : D$UFRWFIO

Syntax : $VINIT ***(start, size, groupSize, numQueues)***
Result : *D$CCODE*

Entry 1 :	***start***	^ BYTE;	Start of buffer descriptor pool & Hash queue
Entry 2 :	***size***	UNSIGNED;	Hash queue length to maintain
Entry 3 :	***groupSize***	BYTE;	Base I/O size in sectors
Entry 4 :	***numQueues***	BYTE;	Number of Hash queues

If Error: **No Error Occurs**
SPRM Ref: VINIT$ Vol.2 Sec. 13.4.3

$VMAPPSK FUNCTION $VMAPPSK

Donate a PSK to Virtual Management

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$VMAPPSK** ***(psk)***
Result : *D$CCODE*

Entry 1 : ***psk*** BYTE;

If Error: IF **$VMAPPSK (...) && D$CFLAG**
THEN *(no more buffer descriptor space);*
SPRM Ref: VMAPPSK$ Vol.2 Sec. 13.4.6

$VPUTBUF FUNCTION $VPUTBUF

Return a Buffer Group to Virtual Pool

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$VPUTBUF** ***(psk, startPage)***
Result : *D$CCODE*

Entry 1 : ***psk*** BYTE;
Entry 2 : ***startPage*** BYTE; page from $VGETBUF

If Error: IF **$VPUTBUF (...) && D$CFLAG**
THEN *(PSK / page was invalid or not found);*
SPRM Ref: VPUTBUF$ Vol.2 Sec. 13.4.8

Establish Memory Window Areas, Virtual

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : **$VSETWIN** ***(winMsnTbl)***
Result : *D$CCODE*

Entry 1 : ***winMsnTbl*** **^ BYTE;** MSN list, terminated 0377

If Error: **No Error Occurs**
SPRM Ref: VSETWIN$ Vol.2 Sec. 13.4.4

Wait for any FAV Operation to Complete

```
Category: System Call
Entered : 82 Jul 01                  Updated : 84 Aug 08   |*
File    : D$RMSIO

Syntax  : $WAITIO   (&fav, wait)
Result  : D$CCODE

Entry 1 : fav    UNSIGNED; a selected FAV                  |*
Entry 2 : wait   BOOLEAN;
* Exit 1: fav    UNSIGNED; returned if more than one       |*
                           FAV was complete.               |*

If Error: IF  $WAITIO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$WAITIO
          $ERRC.$CODE = $ECSI020
Remarks : IF any FAV was a higher number than the          |*
          specified FAV, then the lowest of those is       |*
          returned.  IF all FAVs were lower than the       |*
          specified FAV, then the lowest of those is       |*
          returned.  This is useful for complicated        |*
          round-robin (FMT) schemes.                       |*
SPRM Ref: $WAITIO       Vol.4 Sec.  7.12
SPRM Ref: $WAITIO       Vol.4 Sec.  8.5.6
SPRM Ref: $WAITIO       Vol.4 Sec.  9.1.2.4
SPRM Ref: $WAITIO       Vol.4 Sec.  9.2.2.4
SPRM Ref: $WAITIO       Vol.4 Sec.  9.4.2.6
SPRM Ref: $WAITIO       Vol.4 Sec. 10.3.6
```

Wait for Selected Status Bit Change

Category: System Call
Entered : 83 Apr 02 *Updated* : 84 Jul 25 |*
File : D$RMSIO

Syntax : **$WAITIOS** ***(&fav, &status, wait)***
Result : *D$CCODE*

Entry 1 : ***fav*** **UNSIGNED;** selected FAV |*
Entry 2 : ***status*** **$WSTAT;** selected bits |*
Entry 3 : ***wait*** **BOOLEAN;** Select Alt SC if true
** Exit 1*: ***fav*** **UNSIGNED;** see $WAITIO |*
** Exit 2*: ***status*** **$WSTAT;**

If Error: **IF $WAITIOS (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$WAITIOS
$ERRC.$CODE = $ECSIO20 |*

Workstation, Get Configuration

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSWS

Syntax : **$WCONFIG** ***(&config, &right, &top)***
Result : *D$CCODE*

* *Exit 1*:	***config***	**$WSCONF**; WS kind and config info
* *Exit 2*:	***right***	**BYTE**; Right most possible cursor($WSRC
* *Exit 3*:	***top***	**BYTE**; Top most possible cursor

If Error: **No Error Occurs**
Remarks : See notes in structure $WSCONF.
Screen Sizes:
$WSBL constant is 11, defining the bottom line.
Screens with more than 12 lines will have a negative number in "top".
$WSLC constant is 0, defining left most cursor.
$WSRC constant is 79, defining the minimum right most cursor position (it may be larger).
See Also: $WSTATUS and structure $WSCONF for notes
SPRM Ref: $WCONFIG Vol.4 Sec. 11.3

$WFCLOSE FUNCTION $WFCLOSE

Work-File, Terminate Processing of Work File

```
Category: User Function Routine
Entered : 82 Jul 01                 Updated : 82 Dec 01
File    : D$UFRWFIO

Syntax  : $WFCLOSE  (fcb)
Result  : D$CCODE

Entry 1 : fcb  ^ $WFCB;

If Error: IF  $WFCLOSE (...) && D$CFLAG THEN $ERMSG ();
             $ERRC.$FUNC = SC$SECW,SC$SECWAIT or
                           SC$SECEOF
             $ERRC.$CODE = (see $SECW,$SECWAIT,$SECEOF)

SPRM Ref: WFCLOSE$      Vol.2 Sec. 13.2.3
```

$WFFLUSH FUNCTION $WFFLUSH

Dump Pending Write Buffers to Disk

```
Category: User Function Routine
Entered : 83 Apr 02                 Updated : 83 May 03
File    : D$UFRWFIO

Syntax  : $WFFLUSH  (fcb)
Result  : D$CCODE

Entry 1 : fcb  ^ $WFCB;

If Error: IF  $WFFLUSH (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$ ?
          $ERRC.$CODE = ?
```

$WFOPEN FUNCTION $WFOPEN

Work-File, Prepare an Open Work File For Access

```
Category: User Function Routine
Entered : 82 Jul 01                  Updated : 82 Dec 01
File    : D$UFRWFIO

Syntax  : $WFOPEN  (fcb, openpt, recSize)
Result  : D$CCODE

Entry 1 : fcb      ^ $WFCB;
Entry 2 : openpt   ^ $OPENPT; of $OPEN'ed file
Entry 3 : recSize    UNSIGNED; Record size

If Error: No Error Occurs
SPRM Ref: WFOPEN$       Vol.2 Sec. 13.2.2
```

$WFOPENP FUNCTION $WFOPENP

Open Using Specified Physical I/O Routine

```
Category: User Function Routine
Entered : 83 Apr 02                  Updated : 83 May 03
File    : D$UFRWFIO

Syntax  : WFOPENP  (fcb, openpt, recSize)
Result  : D$CCODE

Entry 1 : fcb       $WFCB;
Entry 2 : openpt ^ $OPENPT; of $OPEN'ed file
Entry 3 : recSize  UNSIGNED; Record size

If Error: IF  $WFOPENP (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$ ?
          $ERRC.$CODE = ?
```

$WFPOSEF FUNCTION $WFPOSEF

Work-File, Position to File's EOF

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$UFRWFIO

Syntax : $WFPOSEF *(fcb)*
Result : *D$CCODE*

Entry 1 : ***fcb*** ^ $WFCB;

If Error: **No Error Occurs**
Remarks : Moves file's EOF pointer to CURRENT
SPRM Ref: WFPOSEF$ Vol.2 Sec. 13.2.5

$WFPOSIT FUNCTION $WFPOSIT

Work-File, Position To File Pointer

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 01
File : D$UFRWFIO

Syntax : $WFPOSIT *(fcb, fp)*
Result : *D$CCODE*
Assign : ***varRslt*** := $WFPOSIT *(fcb, fp)*; |*
IF ***varRslt*** && D$CFLAG=0 & ***varRslt*** && D$ZFLAG
THEN *(pointer given is current EOF);*

Entry 1 : ***fcb*** ^ $WFCB;
Entry 2 : ***fp*** ^ $FILEPTR;

If Error: IF ***varRslt*** && D$CFLAG
THEN *(pointer given is beyond EOF);*
SPRM Ref: WFPOSIT$ Vol.2 Sec. 13.2.6

Work-File, Prepare a New Work File For Access

```
Category: User Function Routine
Entered : 82 Jul 01              Updated : 84 Aug 08   |*
File    : D$UFRWFIO

Syntax  : $WFPREP  (fcb, openpt, recSize)
Result  : D$CCODE

Entry 1 : fcb      ^ $WFCB;
Entry 2 : openpt   ^ $OPENPT; of $OPEN'ed file
Entry 3 : recSize    UNSIGNED; Record size

If Error: IF $WFPREP (...) && D$CFLAG THEN $ERMSG ();  |*
See also: $SEC... and $OPENENV system calls for error  |*
          codes.                                       |*
SPRM Ref: WFPREP$       Vol.2 Sec. 13.2.1
```

Prepare Using Specified Physical I/O Routine

Category: User Function Routine
Entered : 83 Apr 02 *Updated* : 84 Aug 08 |*
File : D$UFRWFIO

Syntax : **$WFPREPP** ***(fcb, openpt, recSize)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**
Entry 2 : ***openpt*** **^ $OPENPT;** of $OPEN'ed file
Entry 3 : ***recSize*** **UNSIGNED;** Record size

If Error: **IF $WFPREPP (...) && D$CFLAG THEN $ERMSG ();**
See also: $SEC... and $OPENENV system calls for error |*
codes. |*

Work-File, Read a Logical Record

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 84 Jul 25   |
File    : D$UFRWFIO

Syntax  : $WFREAD  (fcb, rec)
Result  : D$CCODE
Assign  : varRslt := $WFREAD  (fcb, rec);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG=0
          THEN (record delivered);
          IF varRslt && D$CFLAG=0 & varRslt && D$ZFLAG
          THEN (file is at EOF);

Entry 1 : fcb  ^ $WFCB;
Entry 2 : rec  ^ BYTE; output record area

If Error: IF varRslt && D$CFLAG THEN $ERMSG ();
See Also: $SEC... for error codes                        |
Remarks : Reads logical record from CURRENT,
          advances CURRENT.
SPRM Ref: WFREAD$       Vol.2 Sec. 13.2.7
```

Work-File, Read in LOCATE Mode

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$UFRWFIO

Syntax : **$WFREADL** ***(fcb, &rec)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**
* *Exit 2*: ***rec*** **^ BYTE;** start of record in buffer

If Error: **IF $WFREADL (...) && D$CFLAG THEN $ERMSG ( );**
See Also: $SEC... for error codes |*
SPRM Ref: WFREADL$ Vol.2 Sec. 13.2.10

Name of EXTERNAL RMS UFR used by $WFUPDATE

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

$WFUPDATE FUNCTION $WFUPDATE

Work-File, Update a Logical Record

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 25
File : D$UFRWFIO

Syntax : **$WFUPDATE** ***(fcb, rec)***
Result : *D$CCODE*

Entry 1 : ***fcb*** ^ $WFCB;
Entry 2 : ***rec*** ^ BYTE; input record area

If Error: IF **$WFUPDATE (...) && D$CFLAG THEN $ERMSG (**
See Also: $SEC... for error codes
SPRM Ref: WFUPDAT$ Vol.2 Sec. 13.2.9

$WFUPDATEL FUNCTION $WFUPDATEL

Work-File, Update a Record in LOCATE Mode

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 25 |*
File : D$UFRWFIO

Syntax : **$WFUPDATEL** ***(fcb, &rec)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**
** Exit 2*: ***rec*** **^ BYTE;** Start of record in buffer

If Error: **IF $WFUPDATEL (...) && D$CFLAG THEN $ERMSG ();**
See Also: $SEC... for error codes |*
SPRM Ref: WFUPDTL$ Vol.2 Sec. 13.2.12

WFUPDTL$ FUNCTION WFUPDTL$

Name of EXTERNAL RMS UFR used by $WFUPDATEL

Category: Cross Reference
Entered : 82 Jul 01 *Updated* : 82 Dec 01

$WFWEOF FUNCTION $WFWEOF

Work-File, Write EOF At Current File Position

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 May 03
File : D$UFRWFIO

Syntax : **$WFWEOF** ***(fcb)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**

If Error: **IF $WFWEOF (...) && D$CFLAG THEN $ERMSG ();**
$ERRC.$FUNC = SC$SECW,SC$SECWAIT or SC$SECR
$ERRC.$CODE = (see $SECW,$SECWAIT,$SECR)
Remarks : Saves CURRENT file pointer into EOF and sets EOF pending flag. No actual System Call will occur until $WFCLOSE is called.
SPRM Ref: WFWEOF$ Vol.2 Sec. 13.2.4

$WFWRITE FUNCTION $WFWRITE

Work-File, Write a Logical Record

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 84 Jul 25
File : D$UFRWFIO

Syntax : **$WFWRITE** ***(fcb, rec)***
Result : *D$CCODE*

Entry 1 : ***fcb*** **^ $WFCB;**
Entry 2 : ***rec*** **^ BYTE;** Output record data area

If Error: **IF $WFWRITE (...) && D$CFLAG THEN $ERMSG ()**
See Also: $SEC... for error codes
SPRM Ref: WFWRITE$ Vol.2 Sec. 13.2.8

$WFWRITEL FUNCTION $WFWRITEL

Work-File, Write a Record in LOCATE Mode

```
Category: User Function Routine
Entered : 82 Jul 01                Updated : 84 Jul 25  |*
File    : D$UFRWFIO

Syntax  : $WFWRITEL (fcb, rec)
Result  : D$CCODE

Entry 1 : fcb    ^ $WFCB;
Entry 2 : rec  ^ ^ BYTE; Start of record in buffer

If Error: IF  $WFWRITEL (...) && D$CFLAG THEN $ERMSG ();
See Also: $SEC... for error codes                       |*
SPRM Ref: WFWRITL$      Vol.2 Sec. 13.2.11
```

WFWRITL$ FUNCTION WFWRITL$

Name of EXTERNAL RMS UFR used by $WFWRITEL

```
Category: Cross Reference
Entered : 82 Jul 01                Updated : 82 Dec 01
```

WHILE FUNCTION WHILE

Part of LOOP WHILE Execution Control

Category: DASL Control Word
Entered : 82 Jul 01 *Updated* : 82 Dec 01

Syntax : **see *LOOP function***
DASL Doc: 73,74 March 1982

$WIPEBT FUNCTION $WIPEBT

Clear an Area of Memory to SPACES

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRGEN

Syntax : **$WIPEBT** ***(dest, length)***
Result : *none*

Entry 1 : ***dest*** **^** BYTE; Start of area to be cleared
Entry 2 : ***length*** **UNSIGNED**; Length to be cleared

If Error: **No Error Occurs**
See Also: $WIPEBTA for clear to constant value
SPRM Ref: WIPEBT$ Vol.2 Sec. 8.2

$WIPEBTA FUNCTION $WIPEBTA

Clear an Area of Memory to Constant Value

Category: User Function Routine
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$UFRGEN

Syntax : **$WIPEBTA** ***(dest, length, value)***
Result : *none*

Entry 1 :	***dest***	**^** BYTE;	Start of area to get "value"
Entry 2 :	***length***	**UNSIGNED;**	Length of area
Entry 3 :	***value***	**BYTE;**	Value to be placed in area

If Error: **No Error Occurs**
See Also: $WIPEBT for claer to spaces
SPRM Ref: WIPEBTA$ Vol.2 Sec. 8.2

$WSCTL FUNCTION $WSCTL

Workstation Control Code Function, Used Indirectly

Category: System Call
Entered : 82 Jul 01 *Updated* : 83 Jul 10
File : D$RMSWS

Syntax : **DASL** ***FUNCTION DEFINITION MACRO***

Remarks : Passes parameters to SYSTEM CALL (4) $WSCTL
The control codes used with $WSCTL to
implement the macros below are:
$WSCTLBP, $WSCTCK, $WSCTLCF, $WSCTLCN.
See Also: $BEEP, $CLICK, $CURSOFF, $CURSON for
direct usage of workstation control codes.

Workstation, Obtain One Keyboard Buffer Character

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Jul 01 |*
File : D$RMSWS

Syntax : **$WSGETCH** ***(&char)***
Result : *D$CCODE*

* *Exit 1*: ***char*** CHAR; Character from keyboard

If Error:
```
IF  $WSGETCH (...) && D$CFLAG
 THEN  {
    IF $ERRC.$CODE = $ECWSGCO                         |*
     THEN {Kbd Buffer Empty}
     ELSE $ERMSG ( ) /* $ECWSGC1 */
     };

$ERRC.$FUNC = SC$WSGETCH
$ERRC.$CODE = $ECWSGCO keyboard buffer empty
              $ECWSGC1 workstation offline
```

Remarks : $WSTATUS should be used to determine if the keyboard is character ready ($WSRDY true) so as to avoid error, also to see static status of function keys (if $WCONFIG indicates static status available).
The "char" value returned has not been passed through the user's translate table but control keys have been converted to special or standard codes.
There is a 63 character FIFO queue between the keyboard and the keyboard entry function. Function key upstroke and downstroke codes go through the queue, but the static status indications are real time.

.... continued

Generating Codes on 6600 w/o F-Keys:

```
          $WSF1K thru $WSF5K: digit 1-5 with DISPLAY
          $WSATTK: digit 0 key with DISPLAY key down
          $WSINTK: period key (.) with DISPLAY
          Note: above, downstroke codes are generated.
                The static bits stay true while
                DISPLAY key is down (see $WSTATUS).
                No upstroke codes are generated.
```

See also: LISTS section: $WS... Workstation Keyboard |*
Codes. |*

SPRM Ref: $WSGETCH   Vol.4 Sec. 11.5

Perform Workstation I/O

```
Category: System Call
Entered : 82 Jul 01                  Updated : 84 Jul 01
File    : D$RMSWS

Syntax  : $WSIO    (&mode, string, &hor, &ver, &end)
Result  : D$CCODE

Entry 1 : mode      $WSIOMODE;Initial mode bits: $WSM..
Entry 2 : string ^ CHAR; Data and/or Control Codes
                         Terminated with $ES or $EL
Entry 3 : hor       BYTE; Initial hor cur pos ($WSLC)
Entry 4 : ver       BYTE; Initial ver cur pos ($WSBL-x)
* Exit 1: mode      $WSIOMODE; Mode, unless error:
                              then invalid
* Exit 3: hor       BYTE; Position after last char
* Exit 4: ver       BYTE; Position after last char
* Exit 5: end    ^ CHAR; Char after string terminator

If Error: IF  $WSIO (...) && D$CFLAG THEN $ERMSG ();
          $ERRC.$FUNC = SC$WSIO
          $ERRC.$CODE = $ECUMAV,$ECWSI01,$ECWSI02,
                $ECWSI03,$ECWSI04,$ECWSI05,$ECWSI06

Remarks :     $WSIO Mode (bit) Codes:
            $WSMNW   0001  Inhibit 'DISPLAY' key wait
            $WSMES   0002  Echo secret (char)
            $WSMNI   0004  No case inversion
            $WSMNE   0010  No echo or cursor
            $WSMKCON 0020  Keyin continuous
            $WSMDIGO 0040  Digits only
            $WSMPADN 0100  Pad numeric decimal part
            $WSMNESC 0200 No escape b4 0-037 or 0177
          Initial mode may be changed in "string".
```

$WSIO String Data Char Range:

```
          0 thru $WSIOFCF-1. If $WSMNESC mode bit
          set, 0 thru 040 be interpreted as control
          codes and will not be displayed.
```

$WSIO String Control Code Range:

$WSIOFCF thru $WSIOFCL

$WSIO CONTROL CODES

Note: Some of the control codes require a one byte value following the code, indicated (). Some require a two byte value, indicated (()). Some require some of each.

CURSOR CONTROL

$H	0234	New cursor column follows (pos)
$V	0235	New cursor row follows (pos)
$HA	0236	Cursor column adjustment follows (adj)
$VA	0237	Cursor row adjustment follows (adj)
$CP	0240	Cursor position follows (vert),(horz)
$HU	0241	Home up to upper left-hand corner
$HD	0242	Home down to lower left-hand corner
$NL	0243	Advance to new line
$WSCURON	0214	Turn Cursor On at current position
$WSCUROF	0215	Turn Cursor Off at current position

GENERAL

$EEOF	0200	Erase from cursor to end of frame
$EEOL	0201	Erase from cursor to end of line
$RU	0202	Roll screen up one line
$RD	0203	Roll screen down one line
$ES	0231	End of string
$EL	0232	Advance to new line,terminate strg
$ESNF	0271	End Of String, don't flush display
$NS	0233	New string address follows ((loc))
$WSNOP	0244	No operation
$WSSMODE	0245	Set mode (bits)
$WSCMODE	0246	Clear mode (bits)
$WSCKF	0247	Clear keyboard fifo
$WSBEEP	0204	Beep
$WSCLICK	0205	Click
$WSIKCON	0210	Key click on
$WSIKCOF	0211	Key click off

..... continued

```
$WSATTEN  0216, Enable KDS 3 Attributes;
          underline & 2-level video on 8600 console.
          Has no effect on other workstations.

$WSCONFD 0272  WS Config data (Len),((Loc))
$WSRECON 0273  WS Reconfig data (Len),((Loc))
               where ((Loc)) has the format:
               Mask_0, Value_0, Mask_1, Value_1
               Mask_(Len-1), Value_(Len-1)
```

VIDEO

```
$WSVI    0206  Video inverted
$WSVN    0207  Video normal
$WSSVMOD 0264  Set video mode (mode)
                   VIDEO MODES follow $WSSVMOD:
    $WSVM2L  0000  Vid Mode: Bold-face, double intensit
    $WSVMUNL 0001  Video Mode: Underline
    $WSVMBNK 0002  Video Mode: Blink
    $WSVMAF  0003  Video Mode: Alternate font

    NOTE: Video mode is set to NORMAL at beginning
          of each $WSIO function. The last $WSSVMOD
          function in any string sets the mode for
          the entire string, therefore, only one
          mode can be effected per string.
```

PRINTER

```
$WSPTRON 0212  Turn On Printer connected to WS
$WSPTOFF 0213  Turn Off Printer connected to WS

$WSLF    0265  Line feed for WS serial printers
$WSFF    0266  Form feed for WS serial printers
$WSCR    0267  Carriage return WS serial printers
```

INSERT, DELETE, OPEN, CLOSE, and SCROLL WINDOW

NOTE: in the current RMS workstation, sub window functions work only if the tube is configured for sub windows (in which case the scroll window does N O T work). The roll/scroll window mode will be phased out and all programs should use the sub-window mode.

```
 $WSRESET 0217  Reset Window to Default Screen size
 $WSRESTR 0225, Same as $WSRESET except
          8600 KDS attributes are not disabled

 $WSSWTB  0222  Set sub window (vert-top),(vert-bot)
 $WSSWLR  0223  Set sub window (horz-left),(horz-right)

 $WSIDOCS 0224  (IмогNS, DEL, OPEN, CLOSE, SCROLL codes)

                    Codes following $WSIDOCS:
   $WSINSCH 0000 Insert space under cursor, shift down
   $WSDELCH 0001 Delete char under cursor, shift up
   $WSINSLN 0002 Roll down lines from cursor to bottom
   $WSDELLN 0003 Delete line under cursor and roll up
   $WSOPENL 0004 Open line from under cursor rolling
   $WSCLOSL 0005 Close line from under cursor rolling
```

The scroll commands are followed by the characters that are to make up the column to be scrolled onto the screen. In the current RMS workstation, this string of characters can have imbedded video modifications. $WSIO does not check if other functions are tried.

```
   $WSSCRL  0006 Scroll left < followed by data >
   $WSSCRR  0007 Scroll right < followed by data >
   $WSSCRE  0010 End of scroll data
```

.....continue

OUT and IN Strings

$WSOS	0250	Output string (len),((loc))
$WSONCH	0251	Output repeated (char),(n) times
$WSIS	0252	In string (con),(max),((loc)),(end)
$WSISI	0253	In string imm (con),(max),(skip),(end)
$WSIN	0254	In numeric (lmax),(rmax),((loc)),(end)
$WSINI	0255	In numrc imm (lmax),(rmax),(skip),(end)
$WSITIME	0256	Set inter-char timeout to (t) seconds

KEYIN TRANSLATE TABLE:

Least sig 7 bits = $WSCURS for terminate $WSIO.
Sign bit on characters less than 0100 for terminate |
$WSIO on Control Chars between 0200 and 0277. |
Sign bit on Characters greater than or equal to |
0100 for character invertable by logical XOR with |
040. |

$WSSKXTA	0257	Set keyin translate table at ((loc))
$WSSKXTP	0261	Set keyin xlate table at ((loc)),(psk)
$WSKEYCH	0270	Keyin un-xlated character at ((Loc))
$WSK1CHR	0276	Keyin un-xlated character at ((LOC)) with cursor on/off
$WSECHOS	0262	Set echo secret disply char (char)
$WSTWAIT	0263	Perform n second wait (n)

CHARACTER FONT SET:

The character font set is denoted by a table that is identical to that used by DOS.

$WSLCFS	0260	Load char font set from ((loc))
$WSCURDF	0275	Return to default cursor font
$WSCURFL	0274	Load cursor font from ((Loc))

The $WSLCFS control code causes the character font set for the screen to be loaded from [loc] if the capability exists.

The Character Font table comprises six bytes per entry. The first byte represents the character that is to be displayed on the screen (as a 5 by 7 dot matrix character in this example). The remaining five bytes contain vertical dot positions. One entry in the table might look like this entry for the letter A:

```
 char   (byte 1    byte 2   byte 3   byte 4   byte 5)

 BITS   01234567
CHAR    11000001 [ binary representation of char A ]
byte 1  00111111 [ byte descriptor in binary ]
byte 2  01001000 [ byte descriptor in binary ]
byte 3  01001000 [ byte descriptor in binary ]
byte 4  01001000 [ byte descriptor in binary ]
byte 5  00111111 [ byte descriptor in binary ]
```

The value 11000001 represents the octal value of A with the sign bit set.

When the bits of the five bytes are listed horizontally, with the 1 bits representing the dots in a matrix, the result looks like this:

```
00000    from bit 7 of each byte          [when the
01110    from bit 6 of each byte     xxx    zeros are
10001    from bit 5 of each byte    x   x   replaced
10001    from bit 4 of each byte    x   x   by blanks
11111    from bit 3 of each byte    xxxxx   and the
10001    from bit 2 of each byte    x   x   1's by
10001    from bit 1 of each byte    x   x   x's, the
10001    from bit 0 of each byte    x   x   char A
                                            results]
```

If the first byte in the group (that represents the character to be translated) does not have the sign bit set, an $ECWSI06 error is returned.

Hint: Using the RMS Utility CHAREDIT, the octal string for a character may be verified.

See also: $WCONFIG for screen size.

.... continued

$WSIO ESCAPE SEQUENCE 1 codes
for the Extended Function Keyboard

```
=====================================================
$WSESC1  0226 indicates that $WSIO 'escape-sequence-1' |*
              codes follow.                            |*

Last control code values (for testing):                |*
  $WSESC1L 1, $WSEFCTL;Last escape-seq-1 control code. |*
  $WSEFCL  016, $WSEFST2;Last EFK Control code value.  |*

These codes may follow $WSESC1:                        |*
  $WSDSCNT 0 Disconnect datastation                    |*
  $WSEFCTL 1 Expanded Function Keyboard control.       |*
             See the Escape-Sequence-1 codes that      |*
             follow this code; below...                |*


-----------------------------------------------------
```

Escape 1 Sequences that may be used via:

- **Keycode Translate Module path, or**
- **General Purpose Keyboard Emulation path.**

```
All sequences begin: $WSESC1,$WSEFCTL,...
-----------------------------------------------------

...$WSEFRST 0 Same as $WSRESET except reset all but    |*
              Keycode Xlate Module path                |*
                                                       |*
...$WSEFLOW 1 followed by (CONTROL);                   |*
              Expanded function KBD Flow control       |*
                                                       |*
   CONTROL bit definitions:                            |*
    $WSLENON  001 Enable Keycode Translate Module      |*
                  path; nicknamed 'LENS'               |*
    $WSPUPOF  002 Disable Locked Key Processing;       |*
                  nicknamed 'PUPIL'                    |*
    $WSDKPOF  004 Disable Dead Key Processor           |*
    $WSBIFDT  010 Get data from Internal Buffer        |*
                  Module; nicknamed 'BIFOCAL'          |*
    $WSLNTOF  020 Disable Keycode Translate Module     |*
                  path; nicknamed 'LENS'               |*
```

```
.$WSEFKID 3 followed by ((LOC)); Get keyboard I.D.   |*
.$WSEFSTC 7 followed by ((LOC)); Get current static  |*
          bits, i.e. control codes.                  |*

          First six bytes of LOC contain immediate   |*
          state of the keyboard's control keys.      |*
          7th byte has status of $WSRDY and $WSONL.  |*
          See: Boxes below.                          |*

          Counterpart to $WSTATUS system call.       |*

.$WSEFSTF 010 followed by ((LOC)); Get control key   |*
          static bits from static key FIFO.          |*
          LOC is six bytes long.                     |*
          See: Boxes below.                          |*

          Use only after receiving $WSEFKFO          |*
          (FIFO overflow).                           |*

          Counterpart to $WSTATUS system call        |*
          alternate mode.                            |*

.$WSEFSTL 011  followed by ((LOC)) Get latched       |*
          static bits, i.e. the cumulative result of |*
          any keys pushed since last occurance of    |*
          this control sequence.                     |*

          See $WSEFSTC for contents of LOC's 7 bytes.|*

.$WSEFST2 016 followed by ((LOC)); Get status bits   |*
          + $WSTATUS bits. LOC is nine bytes; the    |*
          same 7 bytes for $WSEFSTC plus the 2 bytes |*
          normally returned by $WSTATUS.             |*

          Note: $WSEFST2 is nearly equivalent to     |*
          calling $WSEFSTC and $WSTATUS but insures  |*
          that all status bits are sampled           |*
          simultaneously.                            |*
                                                     |*
```

.... continued

BYTES 0 THRU 5 : (Returned by $WSEFSTC, $WSEFSTF, and $WSEFSTL)

These six bytes are a bit vector for the 48 possible control keys available on the extended function keyboard. The bits in this vector are numbered starting with 0 (for byte 0, bit 0) to 060 (for byte 5, bit 7).

To find which bit corresponds to a particular control code, subtract 0200 from the control code and divide the result by 2 (to eliminate upstroke codes).

Examples :

 $WSSYSD and $WSSYSU correspond to bit 0
 (byte 0, bit 0)
 $WSTRIAD and $WSTRIAU correspond to bit 023 (19)
 (byte 2, bit 3)

```
...$WSEFECI 5 Enable case inversion                      |*
...$WSEFDCI 6 Disable case inversion                     |*

     Note: $WSEFECI and $WSEFDCI will result in a LOCK |*
     key upstroke or downstroke being sent to the user |*
     program, depending on the state of case inversion.|*

...$WSEFKAB 012 followed by (CKEY); Abort field keyin  |*
            on detection of ocntrol key downstroke.    |*

            This control sequence may be issued for    |*
            more than one key for a cumulative effect. |*

...$WSEFKAR 013  Reset all field keyin abort keys.     |*

...$WSEFKTP 014  followed by (CKEY); Activate function |*
            trap for control key downstroke specified. |*

            This control sequence may be issued for    |*
            more than one key for a cumulative effect. |*

            Note: $TRAPFK is used to set trap address. |*

...$WSEFKTR 015  Reset all function key traps.         |*
```

SPRM Ref: $WSIO   Vol.4 Sec. 11.10

$WSTATUS FUNCTION $WSTATUS

Workstation, Get Current Status Bits

Category: System Call
Entered : 82 Jul 01 *Updated* : 84 Aug 08 |
File : D$RMSWS

Syntax : **$WSTATUS** ***(&status, logical)***
Result : *D$CCODE*

Entry 2 : ***logical*** **BOOLEAN;** Use Alternate SC if true; |
gets status from static |
key FIFO |
* *Exit 1*: ***status*** **$WSTAT;** Status bits

If Error: **IF $WSTATUS (...) && D$CFLAG THEN $ERMSG ( );**
$ERRC.$FUNC = SC$WSTATUS
$ERRC.$CODE = $ECWSTA1
Remarks : Availability of the various workstation status
bits may be determined with $WCONFIG function.

See Also: structure $WSTAT for status bits

SPRM Ref: $WSTATUS Vol.4 Sec. 11.4

$WSWAIT FUNCTION $WSWAIT

Datastation, Enable Port and Wait for Character

Category: System Call
Entered : 82 Jul 01 *Updated* : 82 Dec 01
File : D$RMSWS

Syntax : **$WSWAIT** ***()***
Result : *D$CCODE*

If Error: **No Error Occurs**
SPRM Ref: $WSWAIT Vol.4 Sec. 11.2

Following is a complete list of all downstroke codes:

```
-----------------------------------------------------
| 0-$WSSYSD  0200  20-$WSCIRCD 0250  36-$WSALT2D 0310 |
| 1-$WSVIEWD 0202  21-$WSCKULD 0252  37-$WSALT3D 0312 |
| 2-$WSQUITD 0204  22-$WSCKUCD 0254  38-$WSENT2D 0314 |
| 3-$WSPNTD  0206  23-$WSCKURD 0256    -$WSENTD  0314 |
|                                    39-$WSBAK2D 0316 |
| 4-$WSUNDOD 0210  24-$WSCKCLD 0260    -$WSBAKSD 0316 |
| 5-$WSHELPD 0212  25-$WSCKCCD 0262                   |
| 6-$WSCOPYD 0214  26-$WSCKCRD 0264  40-$WSTABD  0320 |
| 7-$WSREMVD 0216  27-$WSCKDLD 0266  41-$WSNPTBD 0322 |
|                                    42-$WSNPE2D 0324 |
| 8-$WSINSTD 0220  28-$WSCKDCD 0270    -$WSNPEND 0324 |
| 9-$WSRECLD 0222  29-$WSCKDRD 0272  43-$WSLOCKD 0326 |
|10-$WSEFF7D 0224  30-$WSCMDD  0274    -$WSCASED 0326 |
|11-$WSEFF8D 0226  31-$WSENT1D 0276                   |
|                                    44-$WSSHFLD 0330 |
|12-$WSEFF1D 0230  32-$WSNPE1D 0300    -$WSSHF1D 0330 |
|13-$WSEFF2D 0232  33-$WSBAK1D 0302  45-$WSSHFRD 0332 |
|14-$WSEFF3D 0234  34-$WSSHF2D 0304    -$WSSHF3D 0332 |
|15-$WSEFF4D 0236  35-$WSSHF4D 0306  46-$WSALTLD 0334 |
|                                      -$WSALT1D 0334 |
|16-$WSEFF5D 0240                    47-$WSALTRD 0336 |
|17-$WSEFF6D 0242                      -$WSALT4D 0336 |
|18-$WSSQARD 0244                                     |
|19-$WSTRIAD 0246                                     |
 ----------------------------------------------------

 ----------------------------------------------------
|BYTE 6 : (Returned only by $WSEFSTC and $WSEFSTL)    |
|                                                     |
|$WSEFRDY  0001      Key ready                        |
|$WSEFONL  0200      'ONLINE' status (always true)    |
 ----------------------------------------------------
```

... continued

```
-----------------------------------------------------
```

Escape 1 Sequences that require that the

o Keycode Translate Module path be enabled.

```
All sequences begin: $WSESC1,$WSEFCTL,...
-----------------------------------------------------

...$WSEFLCD 2 followed by (MASK),(CONTROL)               |
              LCD display segment control                |

    MASK and CONTROL bit definitions (are the same):     |

      $WSLCDLC  0001 LOWER CASE "a" (0/1) (OFF/ON)       |
      $WSLCDUC  0002 UPPER CASE "A" (0/1) (OFF/ON)       |
      $WSLCDDP  0004 DISPLAY (0/1) (OFF/ON)              |
      $WSLCDKD  0010 KEYBOARD (0/1) (OFF/ON)             |
      $WSLCDSQ  0020 SQUARE (0/1) (OFF/ON)               |
      $WSLCDTR  0040 TRIANGLE (0/1) (OFF/ON)             |
      $WSLCDCR  0100 CIRCLE (0/1) (OFF/ON)               |

    Note: If the MASK bit is set, the CONTROL bit is     |
    interpreted, otherwise the CONTROL bit is ignored.   |

...$WSEFRPT 4 followed by (TIME); set repeat key         |
            timeout in increments of 16 milliseconds.    |


 ----------------------------------------------------
|If TIME is..|THEN action taken by Nucleus is...     |
|----------------------------------------------------|
|     0      |IF keycode is ready, THEN return it.   |
|            |IF not, THEN return $WSRPTK.           |
|----------------------------------------------------|
| 1 - 076    |Set timer to go off after (TIME)*16.   |
|            |IF keycode is received before timeout, |
|            |THEN disable timer AND return keycode, |
|            |OTHERWISE, return $WSRPTK after timeout.|
|----------------------------------------------------|
| 077 - 0377 |Disable repeat key timeout.            |
 ----------------------------------------------------
```

.... continued

TYPES AND STRUCTURES

FORMAT HEADINGS

ENTERED.........:	Date the TYPE was added to Dictionary
UPDATED.........:	Date TYPE definition or comments were changed in the Dictionary.
INCLUDE FILE....:	DASL file where this TYPE is defined
USED by FUNCTION:	Functions who use this TYPE argument
USED in TYPE....:	TYPES that use this TYPE as a field
POINTED TO in...:	TYPES that have pointers to this TYPE

Primary DASL Variable Types

All other variable types are constucted from these primary types, using structures, unions and arrays.

BOOLEAN	;DASL scalar data type: 1 byte unsigned
BYTE	;DASL scalar data type: 1 byte unsigned
CHAR	;DASL scalar data type: 1 byte unsigned
INT	;DASL scalar data type: 2 bytes signed
LONG	;DASL scalar data type: 4 bytes signed
UNSIGNED	;DASL scalar data type: 2 bytes unsigned
FUNCTION	;DASL variable type: up to 13 parameters (1,2 or 4 byte scalar values or pointers) and 1 optional scalar result value. The number of parameters may vary with different compilers.

NOTE: **PCR**
the ***Program Communications Region***

is listed here as a TYPE, but it is *not*.

The PCR is an EXTERNALLY DEFINED group of variables.

Library Absolute Element Header Sector

```
ENTERED.........: 82 Jul 01   UPDATED.: 84 Jul 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

TYPDEF $ABSHDR STRUCT {
      $LIBMAP [64] BYTE; Primary MAP First byte
      ENUMV(0,$LIBMT,$LIBPRIV,$LIBSHAR,$LIBTBAD)
      $LIBNAA UNSIGNED; Next Available Address
      $LIBUCS UNSIGNED; Two Bytes Used; Must be set -1
      $LIBEPT UNSIGNED; Entry Point Address
      $LIBNOPG BYTE; Number of Page Groups
                 DEFINE($LIBMXPG,57)                        |*
      $LIBPG1 [$LIBMXPG] UNION {       Page Groups
            STRUCT {
                  $LIBNPAG BYTE; Number of pages
                  $LIBFPAG UNSIGNED; First page
                  };
            $LIBNPAT UNSIGNED; Number of patch blocks
            };
      $LIBXX   [2] BYTE;  Unused
      $LIBSPID $NAMET;  Shared Program I.D.                 |*
      };
```

Aim Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 84 Jul 01
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBA

```
TYPDEF $ACB STRUCT {         AIM control block
       $ACBSTAT  BYTE; AIM status byte
       $ACBIOST  BYTE; AIM I/O status byte
       $ACBCNFG  BYTE; Index config byte
       $ACBACC   BYTE; File access flags ($ACCODES)
       $ACBLUP   CHAR; Lowest upper-case character
       $ACBHUP   CHAR; Highest upper-case character
       $ACBIGNR  CHAR; Don't care character
       $ACBKEYN  BYTE; Number of keys configured
       $ACBKEYL  BYTE; Aggregate key length
       $ACBEXCL  [7] BYTE; Map of excluded key fields
       $ACBINDX  $LSN; LSN of current index header
       $ACBSLOC  ^ CHAR; Pri recd select key location
       $ACBSLGT  BYTE; Pri recd select key length
       $ACBSKEY  [8] BYTE; Pri recd select key
       $ACBPCNT  UNSIGNED; Nbr of primary maps
       $ACBPSEG  BYTE; Nbr of primary segments/sector
       $ACBPSLN  BYTE; Primary segment length
       $ACBELSN  $LSN; LSN of extension index
       $ACBECNT  UNSIGNED; Number of extension maps
       $ACBETXT  [4] BYTE; Start of data expansion
       $ACBCALL  UNSIGNED; AIM user
       $ACBFALS  UNSIGNED; False hits
       $ACBBUFS  BYTE; Nbr of buffers (todo)
       $ACBCLSN  $LSN; Current LSN
       $ACBMLSN  $LSN; LSN of current AIM map
       $ACBBASE  $LSN; Data file base LSN
       $ACBMSTR  BYTE; Number of master keys
       $ACBNMAP  UNSIGNED; Map number
       $ACBMCUR  UNSIGNED; Access map cursor
       $ACBFCUR  BYTE; Free float buffer curso
       $ACBDCUR  $FILEPTR; Data file cursor
       $ACBPMLN  BYTE; Length of primary maps
       $ACBTMAP  [32] BYTE; Current triplet map
```

....continued

```
      $ACBWFDB  $PFDB; Start of the work buffer PFDB
      $X1       [4-1] $PFDBBUF;
      $ACBPFDB  $PFDB; Start of the AIM index PFDB
      $X2       [8-1] $PFDBBUF;
      };
```

$ACCODES TYPE $ACCODES

File Access Codes

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSIO
USED in TYPE....: $FILEKEY $PIPEGENPT $SECURETBL

TYPDEF $ACCODES    File Access Codes
 SET($NEWFILE,$ACREAD,$ACWRIT,$ACATALG,
     $ACREATE,$ACRENM,$ACKILL,$ACSECQ);

DEFINE($ACREPX,$ACSECQ)
DEFINE($ACMAX,($ACSECQ+$ACKILL+$ACRENM+
       $ACREATE+$ACATALG+$ACWRIT+$ACREAD))
```

BOOLEAN TYPE BOOLEAN

DASL scalar data type: 1 byte unsigned

ENTERED.........: 83 Apr 20   UPDATED.:
  *** DASL COMPILER DEFINED, primary data TYPE ***

BYTE TYPE BYTE

DASL scalar data type: 1 byte unsigned

ENTERED.........: 83 Apr 20   UPDATED.:
  *** DASL COMPILER DEFINED, primary data TYPE ***

$CFGEND TYPE $CFGEND

$SCANCFG Keyword List Terminator

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRSCAN
USED in TYPE....: (with $CFGEND)

TYPDEF **$CFGEND BYTE;** $SCANCFG Keyword List Terminator

$CFGHDR TYPE $CFGHDR

$SCANCFG Configuration Header

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRSCAN
USED by FUNCTION: $SCANCFG
```

```
TYPDEF $CFGHDR [13] CHAR;
```

Note: A $CFGHDR type variable is typically followed
      by a $CFGEND, or by 'n' $CFGKEY's and one $CFGEND.

$CFGKEY TYPE $CFGKEY

$SCANCFG Keyword Parameters

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Mar 02
INCLUDE FILE....: D$UFRSCAN
USED in TYPE....: (with $CFGEND)
```

```
TYPDEF $CFGKEY STRUCT {     $SCANCFG Keyword Parameters
   $CFGKEYW [9] CHAR; Keyword
   $CFGFLAG BYTE; Control byte
                  Masks: $CFGNOVM, $CFGLONG, $CFGGTN
                         $CFGVPT, $CFGFND
   $CFGNOKS BYTE; Number of values
   $CFGTYPE BYTE; Type flag
   $CFGVLGT BYTE; Value length
   $CFGADDR ^ BYTE; Output address
   };
```

CHAR TYPE CHAR

DASL scalar data type: 1 byte unsigned

ENTERED.........: 83 Apr 20 UPDATED.:
***** DASL COMPILER DEFINED, primary data TYPE *****

$CVSTBL TYPE $CVSTBL

$CVSTIME Output Area

ENTERED.........: 82 Jul 01 UPDATED.: 83 Apr 23
INCLUDE FILE....: D$UFRGEN
USED by FUNCTION: $CVSTIME

```
TYPDEF $CVSTBL STRUCT {    $CVSTIME Output Area
   $CVSTDOW BYTE; Day of Week   , 0-6 (0=sunday)
   $CVSTMON BYTE; Month of Year , 0-11(0=Jan)
   $CVSTDAY BYTE; Day of Month  , 0-30(0=1st)
   $CVSTYR UNSIGNED; Year       , 1901-2100
   $CVSTJD UNSIGNED; Julian Date (Day of Year),0-365
   $CVSTHH BYTE; Hour of Day    , 0-23
   $CVSTMM BYTE; Minute of Hour ,  0-59
   $CVSTSS BYTE; Second of Minute, 0-59
   };
```

D$CALLF TYPE D$CALLF

Declare Name to be a Callable Function Type Routine

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$INC
POINTED TO in...: $STARTADR
USED by FUNCTION: D$CALL, D$JUMP |*

TYPDEF ***D$CALLF ();*** Declares a Function type with no parameters, and no RESULT. The function may be defined later in the code, or externally.

D$CCODE TYPE D$CCODE

Condition Code Flags

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$INC

```
TYPDEF D$CCODE
            SET(D$CFLAG, D$ZFLAG, D$SFLAG, D$PFLAG);
```

These are the Condition Code assignments which are valid after a system function has been executed. Often the function RESULT value is TYPE D$CCODE.

Data File Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 01
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBA $FCBD $FCBIS

```
TYPDEF $DCB STRUCT  {          Data File Control Block
 $DCBFLG1   BYTE; DFCB flag byte 1
 $DCBFLG2   BYTE; DFCB flag byte 2
 $DCBACC    BYTE; file access flags ($ACCODES)
 $DCBBLK    BYTE; Block size from $MFDBLSZ
 $DCBCLRP   $FILEPTR; Logical record ptr (LSB..MSB)
 $DCBCLFP   $FILEPTR; Curnt (real) file ptr (LSB..MSB)
 $DCBEOFP   $FILEPTR; EOF pointer
 $DCBCBFP   BYTE; Current buffer
 $DCBHBFP   BYTE; High dirty buffer pointer
 $DCBCPFD   ^ $PFDB; Current PFDB pointer
 $DCBAPFD   ^ $PFDB; Alternate PFDB pointer
 UNION {
    STRUCT {
       $DCBPFDB  $PFDB; Data file PFDB
       $X7 [16-1] $PFDBBUF; 15 more buffers
       };
    $DCBMAB    $MAB; Managed DCB redefinition of PFDB
    };
 };
```

$DISTBL TYPE $DISTBL

$DISORT Parameter Table

ENTERED..........: 82 Jul 01   UPDATED.: 83 Jul 23
INCLUDE FILE....: D$UFRGEN
USED by FUNCTION: $DISORT

```
TYPDEF $DISTBL STRUCT {      $DISORT Parameter Table
   $DISNUM UNSIGNED; Number of table entries
   $DISTAB ^ BYTE; Location of table
   $DISRLEN UNSIGNED; Length of table entry
   $DISKEY UNSIGNED; Displacement of key within entry
   $DISKEYL UNSIGNED; Length of key
   $DISWORK ^ BYTE; Location of work area
   };
```

$DSTINFO TYPE $DSTINFO

Daylight Savings Time Start/Stop Table

ENTERED..........: 83 Jul 23   UPDATED.:
INCLUDE FILE....: D$RMSGEN
USED in TYPE....: $SYSTINFO

```
TYPDEF $DSTINFO STRUCT {
   $TMDSTMN BYTE;  Month to Start/End DST
   $TMDSTWD BYTE;  Day of Week to Start/Stop DST
   $TMDSTDC BYTE;  Number of Days to Count
   $TMDSTFG   DST Start/Stop Flags
           SET($TMDSTDI); 1=From End
                          0=From Start
   $TMDSTHR BYTE;  Hour Start/Stop DST
   };
```

$ENVN TYPE $ENVN

Environment Name

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $ENVDEL $ENVLOC
USED in TYPE....: $ENVT $NAMEEXTENV $SFENT
POINTED TO in...: $FILESPK

TYPDEF $ENVN [8] CHAR;     Environment Name
```

$ENVT TYPE $ENVT

User Environment Table Entry

```
ENTERED.........: 83 Apr 02    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$UFRENV
USED by FUNCTION: $ENVFNDM $ENVINS  $ENVLGET $ENVLOC
                  $ENVPDAT $ENVPEEL $ENVPHSI $ENVPNAM
                  $ENVPNET $ENVPNOD $ENVPPAS $ENVPRES
                  $FILEPCN
POINTED TO in...: $ENVT (itself, linking action)

TYPDEF $ENVT STRUCT {     Environment Table Entry
   NAME $ENVN; Environment name
   LINK ^ $ENVT; Link to next entry (set by UFR)
   DATA [0] CHAR; Environment data (Net, Node,
                  HSI, Passwords, Access Codes);
                  See $OPENPT.$OTENV
   };
Note: Blank fill with $WSBLANK characters
```

$ERRCODE TYPE $ERRCODE

RMS Standard Error Code

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMS
USED in TYPE....: $ERRC (an external variable)          |*
```

```
TYPDEF $ERRCODE STRUCT {      RMS Standard Error Code
   $CODE BYTE; Error number
   $FUNC BYTE; Routine number
   };
```

$EXTT TYPE $EXTT

File Extension

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED in TYPE....: $FILEINFO $NAMEEXT $NAMEEXTENV $SFENT
                  $SFITABLE
POINTED TO in...: $FILESPK
```

```
TYPDEF $EXTT [4] CHAR; ;    File Extension
```

```
Note: Blank fill with $WSBLANK characters
      or regular blanks.
```

File Control Block for AIM File Structure

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Mar 02
INCLUDE FILE....: D$FAR
USED by FUNCTION: $ACLOSE  $ADELCR  $AINS    $AIOCLR
                  $AOPEN   $APOS    $AREAD   $AREADCR
                  $AREADKG $ARWRTCR $AWRITE
USED in TYPE....: $FCBAIM

TYPDEF $FCBA STRUCT {
 $FCBFLG1 BYTE; Flag byte 1
                Type Mask: $FCSTMSK
                Types: $FCSTICB, $FCSTDCB, $FCSTSIB,
                $FCSTPRT, $FCSTPRU, $FCSTACB, $FCSTSAB
                Other Bits: $FCSOVER, $FCSCMPR,
                $FCSBIN, $FCSOPEN, $FCSMNGD
 $FCBUREC ^ CHAR; User record address
 $FCBRLGT UNSIGNED; Length of the user record
 $FCBDBFS BYTE; Numbr of buffers assciatd with D.PFDB
 $FCBFLG2 BYTE; AIM flag byte
                Bits: A$ABORT, A$APMISS, A$PRISEL
                      A$UPCASE, A$SHARE
 $FCBKEY  ^ ^ CHAR; Pointer to user's key list
                    See example in $FCBAIM
 $FCBKLGT BYTE; Number of keys in key list
 $FCBBLKL BYTE; Number of AIM PFDB buffers
 $FCBLINK [2] UNSIGNED; Primary links
 $FCBSLLH [2] UNSIGNED; Secondary links
 $FCBHASH BYTE; Hash code for data file name
 $FCBACB  $ACB; Start of the ACB
 $FCBDCB  $DCB; Start of the DFCB (data file)
 };
```

Macro to Configure AIM File Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 83 Apr 12
INCLUDE FILE....: D$FAR

```
DEFINE ($FCBAIM, #[
 STRUCT {
   $FCBA $FCBA;
   }
 IFELSE(#1,,,#[:=$FCBAIMI(#1,#2,#3,#4,#5,#6)#])
 #])
```

```
PROGRAM EXAMPLE:
/* User's Key List
   ---------------
   <length>   number of ASCII bytes in specification
   1          Index to first character
   'NNS<key>' Key specification where:
NN is field:2 decimal digits or blank, decimal digit
S specifies search type (X,L,R,or F)
<key> is the actual key characters

Note: var [] CHAR := ... Automatically counts
                          array size assigned.     */

key1 [] CHAR := {<length>,1,'NNS<key>'};
key2 [] CHAR := {<length>,1,'NNS<key>'};
key  [] ^ CHAR := { &key1[0], &key2[0]}; /* Pointer
                                             Array  */
/* Macro Parameters:
   &record[0] Address of first character in record
   maxrcdln   Max Record Length
   &key[0]    Address of first Key Pointer
   keyln      Number of Keys
   [,rbuf]    Number of Data Buffers [1 if no spec]
   [,ibuf]    Number of Index Buffers[1 if no spec]
                                                   */
 aim file $FCBAIM ( &record[0], maxrcdln, &key[0],
                    keyln [,rbuf] [,ibuf]);
  { aim file.$FCBA.$FCBFLG1 := ? ; ? = some flag
    IF $AIOCLR(&work, &aim file.$FCBA) && D$CFLAG
    THEN $ERMSG( ); }
```

Initializer Macro used by Macro $FCBAIM

ENTERED.........: 83 Apr 12   UPDATED.: 84 Jul 15
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBAIM

```
DEFINE($FCBAIMI,#[
   {{ $FCSTACB,#1,#2,                                      |*
      IFELSE(#5,,2,#5),0,#3,#4,IFELSE(#6,,1,#6),           |*
      {0,0}, {0,0}, 0                                      |*
      { 0,0,0,0,'A','Z','?',0,0,                           |*
        {0,0,0,0,0,0,0}, {0,0},                            |*
        0,0,'????????',0,0,0,                              |*
        {0,0}, 0, {0,0,0,0}, 0,0,0,                        |*
        {0,0}, {0,0}, {0,0}, 0,0,0,0,                      |*
        { 0, {0,0} }, 0,                                   |*
        { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,                   |*
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0                    |*
          },                                               |*
        {  -1,{-1,-1},   4,-1, 4, {{-1,-1,-1}} },          |*
        { {-1, -1,-1}, {-1,-1,-1}, {-1,-1,-1}  },          |*
        {  -1,{-1,-1},   8,-1, 8, {{-1,-1,-1}} },          |*
        { {-1, -1,-1}, {-1,-1,-1}, {-1,-1,-1},             |*
          {-1, -1,-1}, {-1,-1,-1}, {-1,-1,-1}              |*
          {-1, -1,-1},                                     |*
          }                                                |*
        },                                                 |*
      { 0,0,0,0,                                           |*
        { 0, {0,0} },                                      |*
        { 0, {0,0} },                                      |*
        { 0, {0,0} },                                      |*
        0,0,0,0,                                           |*
        {  -1, {-1,-1},   2,-1, 2, {{-1,-1,-1}} },         |*
        { {-1,  -1,-1}, {-1,-1,-1} }                       |*
        }                                                  |*
      }}                                                   |*
   #])                                                     |*
```

File Control Block for Direct File

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Mar 02
INCLUDE FILE....: D$FAR
USED by FUNCTION: $DCLOSE  $DDEL     $DDELCR  $DGETCRK
                  $DIOCLR  $DOPEN    $DPOS    $DPOSEOF
                  $DPOSNX  $DPOSPV   $DREAD   $DREARCR
                  $DREADNX $DREADPV  $DRWRT   $DRWRTCR
                  $DWRITE  $DWRITENX $DWRTEOF
USED in TYPE....: ( Initializer Macros: $FCBDIR
                       $FCBDIRPRT $FCBDOVR $FCBPRT )

TYPDEF $FCBD STRUCT {
 $FCBFLG1 BYTE; Flag byte 1
                Type Mask: $FCSTMSK
                Types: $FCSTICB, $FCSTDCB, $FCSTSIB,
                $FCSTPRT, $FCSTPRU, $FCSTACB, $FCSTSAB
                Other Bits: $FCSOVER, $FCSCMPR,
                $FCSBIN, $FCSOPEN, $FCSMNGD
 $FCBUREC ^ CHAR; User record address
 $FCBRLGT UNSIGNED; Length of the user record
 $FCBDBFS BYTE; Number of buffers associated w/D.PFDB
 $FCBDCB2 $DCB; }; Start of the data control block
```

$FCBDIR TYPE $FCBDIR

Macro to Configure Direct File Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$FAR

```
DEFINE ($FCBDIR, #[ $FCBDIRPRT(#1,#2,#3,$FCSTDCB) #] )
```

see structure macro: $FCBDIRPRT

PROGRAM EXAMPLE: (see parameters in $FCBAIM)

```
 direct_file $FCBDIR (&record[0],maxrcdln[,nbuf]);

  { direct_file.$FCBD.$FCBFLG1 := ?;
    IF $DIOCLR(&work, &direct_file.$FCBD) && D$CFLAG
    THEN $ERMSG(); }
```

$FCBDIRPRT TYPE $FCBDIRPRT

Macro used by Macros $FCBPRT,$FCBDIR,$FCBDOVR

ENTERED.........: 82 Jul 01   UPDATED.: 83 Apr 12
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBDIR $FCBDOVR $FCBPRT

```
DEFINE ($FCBDIRPRT, #[
 STRUCT {
  $FCBD $FCBD;
   IFELSE(#5,,,$pfdb $PFDB; $pfdbb [16-1] $PFDBBUF;)
  }
IFELSE(#1,,,#[:=$FCBDIRPRTI(#1,#2,#3,#4)#])
 #] )
```

EXAMPLES OF USE:

```
DEFINE ($FCBDIR,
     #[ $FCBDIRPRT(#1,#2,#3,$FCSTDCB)#])
DEFINE ($FCBDOVR,
      #[$FCBDIRPRT(#1,#2,#3,$FCSTDCB+$FCSOVER,1)#])
DEFINE ($FCBPRT,
      #[$FCBDIRPRT(#1,#2,#3,$FCSTPRT,1)#])
```

$FCBDIRPRTI TYPE $FCBDIRPRTI

Initializer Macro used by Macro $FCBDIRPRT

ENTERED.........: 83 Apr 12   UPDATED.:
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBDIRPRT

```
DEFINE ($FCBDIRPRTI,
        #[{{#4,#1,#2,IFELSE(#3,,1,#3)}}#])
```

$FCBDOVR TYPE $FCBDOVR

Macro to Configure Direct-Overlapped I/O FCB

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$FAR

```
DEFINE ($FCBDOVR,
     #[ $FCBDIRPRT(#1,#2,#3,$FCSTDCB+$FCSOVER,1) #] )
```

see structure macro: $FCBDIRPRT

PROGRAM EXAMPLE: (see parameters in $FCBAIM)

```
 dir_ovr_file $FCBDOVR (&record[0],maxrcdln[,nbuf]);

  { dir_ovr_file.$FCBD.$FCBFLG1 := ?;
    IF $DIOCLR(&work, &dir_ovr_file.$FCBD) && D$CFLAG
    THEN $ERMSG(); }
```

File Control Block for ISAM File

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Mar 02
INCLUDE FILE....: D$FAR
USED by FUNCTION: $ICLOSE  $IDEL    $IDELCR  $IDELK
                  $IINS    $IIOCLR  $IOPEN   $IPOS
                  $IPOSKP  $IPOSKS  $IPREP   $IREAD
                  $IREADCR $IREADKP $IREADKS $IRWRT
                  $IRWRTCR $IWRITE
USED in TYPE....: $FCBISAM

TYPDEF $FCBIS STRUCT {
 $FCBFLG1 BYTE; Flag byte 1
                Type Mask: $FCSTMSK
                Types: $FCSTICB, $FCSTDCB, $FCSTSIB,
                $FCSTPRT, $FCSTPRU, $FCSTACB, $FCSTSAB
                Other Bits: $FCSOVER, $FCSCMPR,
                $FCSBIN, $FCSOPEN, $FCSMNGD
 $FCBUREC ^ CHAR; User record address
 $FCBRLGT UNSIGNED; Length of the user record
 $FCBDBFS BYTE; Numbr of buffrs associatd with D.PFDB
 $FCBFLG2 BYTE; User ISAM flag byte
                Bits: $FCSIDUP, $FCSISHR, $FCSNKEY
 $FCBKEY  ^ CHAR; Address of the user's ISAM key area
 $FCBKLGT BYTE; Length of the ISAM key
 $FCBBLKL BYTE; Length of ISAM block in sectors
 $FCBLINK [4] BYTE; Primary links
 $FCBSLLH [4] BYTE; Secondary links
 $FCBHASH BYTE; Hash code for data file name
 $FCBICB  $ICB; Start of the ICB
 $FCBDCB1 $DCB; Start of the DFCB (data file)
 };
```

$FCBISAM TYPE $FCBISAM

Macro to Configure ISAM File Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 83 Apr 12
INCLUDE FILE....: D$FAR

```
DEFINE ($FCBISAM, #[
 STRUCT {
  $FCBIS $FCBIS;
  $k [1+ IFELSE(#4,,0,(#4)) ] BYTE;
  }
 IFELSE(#1,,,#[:=$FCBISAMI(#1,#2,#3,#4,#5)#])
 #] )

    PROGRAM EXAMPLE: (see parameters in $FCBAIM )

 isam_file $FCBISAM ( &record[0], maxrcdln, &key[0],
                             keyln [,nbuf] );
  { isam_file.$FCBIS.$FCBFLG1 := ? };
    IF $TIOCLR(&work, &isam_file.$FCBIS) && D$CFLAG
    THEN $ERMSG(); }
```

$FCBISAMI TYPE $FCBISAMI

Initializer Macro used by Macro $FCBISAM

ENTERED.........: 83 Apr 12   UPDATED.:
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBISAM

```
DEFINE($FCBISAMI,
   #[{{$FCSTICB,#1,#2,IFELSE(#5,,1,#5),0,#3,#4,0}}#])
```

Macro to Configure Print File Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$FAR

```
DEFINE ($FCBPRT,
     #[ $FCBDIRPRT(#1,#2,#3,$FCSTPRT,1)#])
```

see structure macro: $FCBDIRPRT

PROGRAM EXAMPLE: (see parameters in $FCBAIM)

```
 direct_file $FCBDIR (&record[0],maxrcdln[,nbuf]);

  { direct_file.$FCBD.$FCBFLG1 := ?;
    IF $DIOCLR(&work, &direct_file.$FCBD) && D$CFLAG
    THEN $ERMSG(); }
```

FFMTABL$ TYPE FFMTABL$

File Format Table

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 01
INCLUDE FILE....: D$UFRGEN

```
EXTERN FFMTABL$ [0] STRUCT{      File Format Table
   $FFTCODE BYTE; File Format Code ($FFMT...)
   $FFTNAME [4] CHAR; File Format Name
   };
```

$FILEINFO TYPE $FILEINFO

Info Returned by $FILENAM Mode of $FILES

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 01
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $$FILENAM
POINTED TO in...: $FILESTBL

```
TYPDEF $FILEINFO STRUCT {
   $FILFNAM $NAMET; File name
   $FILFEXT $EXTT; Extension
   $FILFLEN $LSN; LSN of EOF sector
   $FILFINC UNSIGNED; File increment in sectors
   $FILFFMT BYTE; File format code ($FFMT...)
   $FILFCT $TIME; File creation time
   $FILSEGM BYTE; Number of used segments, 32 max
   $FILALLO $LSN; LSN of last allocated sector
   };
```

$FILEKEY TYPE $FILEKEY

File Key Structure

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSIO
USED in TYPE....: $FILEKEYS

```
TYPDEF $FILEKEY STRUCT {        File Key Structure
   $FDTKEYL $PACKPW; Packed Password
   $FDTACCO $ACCODES; Access Code
   };
```

$FILEKEYS TYPE $FILEKEYS

File Key List Array

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSIO
USED in TYPE....: $PIPEGENPT $SECURETBL

```
TYPDEF $FILEKEYS [$FDTKEYN] $FILEKEY;
            Array of 9 ($FDTKEYN) keys
            (password, access code)
```

File Pointer Structure

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 23
INCLUDE FILE....: D$RMS
USED by FUNCTION: $DDEL     $DGETCRK $DGETNRK
                  $DPOS     $DREAD   $DRWRT
                  $DRWRITE $TXPOSIT $WFPOSIT
USED in TYPE....: $ACB  1 time    $DCB  3 times
                  $ICB  1 time    $WFCB 3 times
```

```
TYPDEF $FILEPTR UNION {  File Pointer Structure
   STRUCT {
      $FPTRBUFOF BYTE; Offset within sector
      $FPTRLSN $LSN; LSN
      };
   $FP LONG; Byte offset within file.
             Total length of file to current position
             in bytes. (Alternate method of file
             analysis).
   };
```

$SCANFLS File Specification

ENTERED.........: 82 Jul 01   UPDATED.: 83 Feb 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $SCANFLS

```
TYPDEF $FILESPK STRUCT {
   $FSOSFT $SFENT; Symbolic file table
   $FSOOPT                       Option
           SET($FILNAMR, $FILEXTR, $FILENVR, $FILANYR,
               $FILFDEF, $FILQMOK, $FILNDSP);
   $FSODNAM ^ $NAMET; Default name pointer
   $FSODEXT ^ $EXTT; Default extension pointer
   $FSODENV ^ $ENVN; Default environment pointer
   };
```

PROGRAM EXAMPLE:

Use of file name scanner:

```
fileSpk [] $FILESPK := {
{ { 'IN      ', <$NAMET>'',<$EXTT>'',      <$ENVN>'' },
  $FILNAMR,      $NOADR,   &<$EXTT>'TEXT', &<$ENVN>'' },
{ { 'OUT     ', <$NAMET>'',<$EXTT>'',      <$ENVN>'' },
  $FILNDSP,      &fileSpk[0].$FSOSFT.$SFTNAM,
                           &<$EXTT>'REL',  &<$ENVN>'' }
};

  IF $SCANFLS(&fileSpk[0], 2) && D$CFLAG THEN $ERMSG();
```

Redefinition of $PFDB, Multi-File System Calls

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $FILES $GETSFI

```
TYPDEF $FILESTBL STRUCT {
   $PFVID UNSIGNED; File access variable                    |*
                    identification from $PFDB               |*
   UNION {
      STRUCT {   ...for $GETSFI                             |*
         $FX1 [6] BYTE; unused                              |*
         $SFIP ^ $SFITABLE; HSI, etc.                       |*
         };
      STRUCT {  Structure for $FILEPCN,$FILENAME mode
         $FPCNP ^ UNSIGNED; Pointer to  PCN's
         $FX2 BYTE;
         $FTODO UNSIGNED; Number of PCN's to convert
         $FX3 BYTE;
         $FNAMP ^ $FILEINFO; File info storage pointr
         };
      STRUCT {   Structure for $FILECHK mode
         $FPCN UNSIGNED; PCN of file
         $FFLAG BOOLEAN; File open flag
         };
      };
   };
```

"FUNCTION" TYPE "FUNCTION"

DASL Function variable data type

ENTERED.........: 83 Jul 23   UPDATED.:

***** DASL COMPILER DEFINED, primary data TYPE *****

The word "Function" is not used to declare function variable types; the parenthesis symbols are used: ().

Example:

```
FuncName (param1 type, ...param13 type) resultType :=
VAR variables types;
{ Defining code };
```

$HSI TYPE $HSI

Hierarchical Structure Information Array

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED in TYPE....: $SFITABLE

TYPDEF ***$HSI*** [32] CHAR;

The high order bit of the last character of each catalog file name is set.

Example:
Catalog HSI seen in use as   A.DOG.DATA
is stored as

```
              ADOGDATA
  Octal MSBits 31131113
              00100020
        LSBits 14774141
```

ISAM Control Block

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 01
INCLUDE FILE....: D$FAR
USED in TYPE....: $FCBIS

```
TYPDEF $ICB STRUCT {        ISAM Control Block
 $ICBFLG1  BYTE; Flag byte 2(see $IFSTB LRIOCDEF/SRC)
 $ICBACC   BYTE; Index file access flags ($ACCODES)
 $ICBLFP   $LSN; LSN of the ISAM LFP (LSB.MSB)
 $ICBTOP   $LSN; LSN of top of tree (LSB.MSB)
 $ICBCURS  ^ BYTE; Current block and offset (LSB,MSB)
 $ICBDCUR  $FILEPTR; Data file cursor  (LSB.MSB)
 $ICBMXKL  BYTE; Maximum key length
 $ICBCKEY  ^ CHAR; Address of the key save area
 $ICBPFDB  $PFDB; Start of the index PFDB
 $X1 [8-1] $PFDBBUF; 7 more buffers
 };
```

24 Bit Number Structure

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$INC

```
TYPDEF ILONG STRUCT {   24 Bit Number Structure
   LSW UNSIGNED; Least significant 16 bits
   MSB BYTE; Most siginificant byte
   };
```

Info Returned by $INFO

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSGEN
USED by FUNCTION: $INFO

TYPDEF $INFOITEM UNION {       Info Returned by $INFO
 STRUCT {                        RESOURCES
  $IRONAME $NAMET; Resource name
  $IROKIND BYTE; Resource kind     ( $DK...)
  $IROSUBK BYTE; Resource sub-kind ( $SK...)
  $IROFLAG $RSRCFLAGS; Resource flags    ( $IRF.. )
  UNION {
     STRUCT {          DISKS,MAG-TAPE,CASSETTES,COMM
      $IROSFTR UNSIGNED; Soft read error counter
      $IROHRDR UNSIGNED; Hard read error counter
      $IROSFTW UNSIGNED; Soft write error counter
      $IROHRDW UNSIGNED; Hard write error counter
      $IRORDCT ULONG; Read activity counter
      $IROWRCT ULONG; Write activity counter
      UNION {
        STRUCT {          DISKs MAG-TAPE, CASSETTEs
         $IROSUBD BYTE; Physical sub-device number
         $IROFREC UNSIGNED; Free clusters (DISK ONLY)
         $IROMAXC UNSIGNED; Max cluster avail ( " )
         $IROSCLU BYTE; Sec. per cluster ( "  )
         $IROFCNT UNSIGNED; Open files counter ( " )
         $IROTIME $TIME; Time of last access ( " )
         $IROCVID UNSIGNED; Contrlr var.serial num(")?
         };
        STRUCT {         COMM DEVICE RESOURCE ONLY
         $IRCDRVR BYTE; Driver overlay ident. number
         $IRCOMET BYTE; Error threshold
         };
        };
      };
     $IROERCT UNSIGNED; Error counter CARD READER
     };
  };                             LINE PRINTER
```

......continued

```
STRUCT {                               CONTROLLERS
 $ICID UNSIGNED; Controller variable identifier

 $ICKIND ENUM    Controller kind ($CK...)
        ($CK9350, $CK9370, $CK9374, $CK9390,
        $CK88D1, $CK88MEM);
        ENUMV(6, $CK9301, $CK9310, $CK1403,
        $CK9315, $CK9324);
 UNION {
    STRUCT {                 $CK9350,9370,9374,9390
       $ICPORT BYTE; Logical port number
       $ICACCNT [4] BYTE; Activity counter,read & wrt |*
       $ICECNT1 UNSIGNED; Error counter 1
       $ICECNT2 UNSIGNED; Error counter 2
       $ICECNT3 UNSIGNED; Error counter 3
       $ICECNT4 UNSIGNED; Error counter 4
       $ICECNT5 UNSIGNED; Error counter 5
       };
    STRUCT {                    $CK88MEM
       $ICMBANK BYTE; Memory bank number of CTV
       $ICMBITS [22] BYTE; 22 bit err counters
                           in CTV
       };
    };
 };
STRUCT {                        TASKS
     $ITONAME $NAMET; Task name
     $ITOMINM BYTE; Minimum number of  sectors
     $ITOACTM BYTE; Actual number of memory sectors
     $ITOID   BYTE; Task identification                |*
     $ITOFATH BYTE; Father task identification
     $ITOPRTY BYTE; Priority 0 - $PRIMAX ($PRINORM)
     $ITOMAXM BYTE; Maximum Number of Memory sectors
     };             Requested (PCR Utility, $MAXMEM)
STRUCT {
     $ISPNAME NAMET; Shared Program Name
     $ISPSTAT BYTE;  Shared Program Status
       DEFINE( $ISPMEM,037)
       SETV( 1<<7,$ISPLOK )
     $ISPUSER BYTE; Shared Program User Count
     };
```

......continued

```
STRUCT {              NODE Startup / Boot Data
     $INSTART $TIME; Startup time
     $INVRP [5] CHAR; Ver/Rev/Pre ASCII lettrs vvrrp
     $INBNLPT BYTE; Nucleus library processor type
     $INBNLSL CHAR; Nucleus library suffix letter
     $INBCLSL CHAR; Command/DLL library suffix lettr
     $INBNETN $NAMET; Boot net name
     $INBNODN $NAMET; Boot node name
     $INBRESN $NAMET; Boot resource name
     };


                 $NODEND
                      PL
STRUCT {                   NODES
     $INETNAM $NAMET; Network name
     $INONAME $NAMET; Node name
     $INFLAG $NODEFLAGS; Flags ( $INF...)
     $INDID BYTE; Destination identification
     };


                 $LINKN
STRUCT {              PL     CONNECTION LINKS
     $ILNAME $NAMET; Network Name
     $ILFLAGS $NODEFLAGS; Flags ( $INF...)
     $ILRXMES ULONG; Received message counter
     $ILTXMES ULONG; Transmitted message counter
     $ILIGCNT UNSIGNED; Ignored received message cnt
     $ILTXERR UNSIGNED; Transmission Error Count
     $ILTXABT UNSIGNED; Transmsn aborted(TA timeout)
     $ILRCONF UNSIGNED; Reconfiguration counter
     };
```

......continued

```
STRUCT {                    RESOURCE UTILIZATION INFO
     $IRUFLAG $IRUFLAGS; Flags ( $IRU.. )
     $IRUTBUF BYTE; Total number of incoming buffers
     $IRUFBUF BYTE; Number of free incoming buffers
     $IRUTFAV BYTE; Total number of incoming FAVs
     $IRUCFAV BYTE; Number of consumed incoming FAVs
     $IRUWFAV BYTE; Num.incomng FAVs waitng on bufrs
     $IRUMEMA UNSIGNED; Nbr of available mem sectors
     $IRUSTFA UNSIGNED; System table first address
     $IRUSTEA UNSIGNED; System table end address
     $IRUMBUF BYTE; Peak value of($IRUTBUF-$IRUFBUF)
     $IRUMFAV BYTE; Peak value of $IRUCFAV
     $IRUMKBC BYTE; Nbr of buffrs usd for FAV markrs
     $IRUMKFC UNSIGNED; Number of free FAV markers
     $IRUOVAC UNSIGNED; Overlay access counter
     $IRUOVLD UNSIGNED; Overlay load counter
     $IRUOVWT BYTE; Overlay wait counter               |*
     $IRUSRWT BYTE; Nbr of users waiting to execute    |*
     };


$DELIMT [16] BYTE; Name delimiter characters
                   (Get info about Delimiters ?)


$INFODUMMY [50] BYTE;
};

Note: See  SPRM Vol. 4 Sec. 3.4 Figure 3-1
      (Info Item Structure)
Note: RMSCDEFS/SRC also provides definition
      of these elements
```

INT TYPE INT

DASL scalar data type: 2 bytes signed

ENTERED.........: 83 Apr 20 UPDATED.:
*** DASL COMPILER DEFINED, primary data TYPE ***

$INTS TYPE $INTS

Interupt State Table

ENTERED.........: 82 Jul 01 UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSPROG
USED by FUNCTION: $RFIAKS $RFIDKS $RFIFK $RFIKKS
$TRAPAKS $TRAPDKS $TRAPFK
$TRAPKKS

```
TYPDEF $INTS STRUCT {
   $INTSCC BYTE; Condition code
   $INTSREG [8] BYTE; Registers
   $INTSRAD ^ BYTE; Return address
   $INTSXAD $STARTADR; Execute address
   };
```

See SPRM Vol. IV Sec 2.4

$IRUFLAGS TYPE $IRUFLAGS

Resource Utilization Flags

ENTERED.........: 82 Jul 01 UPDATED.: 83 Apr 12
INCLUDE FILE....: D$RMSGEN
USED in TYPE....: $INFOITEM

```
TYPDEF $IRUFLAGS
                 SET( $IRUx, $IRUIFH, $IRUOFH, $IRUIFA);
```

$LIBENTRY TYPE $LIBENTRY

Library Directory Entry

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR (an array of $LIBENTRY )

TYPDEF $LIBENTRY STRUCT {
  $LIBMTYP $LIBTYPE; Type of this member:
           See $LIBTYPE structure (names & values)
  $LIBMNAM $LNAMET; Member name
  $LIBMLSN UNSIGNED; First LSN
  $LIBMLEN UNSIGNED; Length in sectors
  $LIBMVRN BYTE; 3-bits version, 5-bits revision
  $LIBMLMD UNSIGNED; Last modification date
  };
```

$LIBSECTOR TYPE $LIBSECTOR

Library Sector Formats

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSSTRUCT

TYPDEF $LIBSECTOR UNION {
  $LIBDIR [16] $LIBENTRY; Library directory
  $ABSSECTOR UNION {          Absolute member sectors
      $LAHDR $ABSHDR; absolute header
      $LACODE [256] BYTE; absolute code
      };                  Use this to analyse a file
                          with unspecified format.
  $RELSECTOR STRUCT {    Relocatable sectors
      $LIBSCODE $RELCODE; Relocatable sector type
      UNION {
       $LRPID $RELPID; Rel program ID sector
       $LROBJ $RELOBJ; Rel object code sector
       $LRXDEF $RELXDEF; Rel external def sector
       $LRXREF $RELXREF; Rel external ref sector
       $LRXFER $RELXFER; Rel starting adr sector
       $LRXEPN $RELEPNS; Rel entry point member sectr
       $LRLINE $RELLINE; Rel DEBUG line numbers sectr
       };
      };   NOTE: This type is used to read and write   |*
  };             library file sectors.                 |*
```

$LIBTYPE TYPE $LIBTYPE

Library Directory Entry Types

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Feb 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBENTRY
```

```
TYPDEF $LIBTYPE
   ENUM($LIBFREE,$LIBTERM,$LIBLINK,$LIBABSX,
        $LIBABSO,$LIBRELL,$LIBT006);
   ENUMV(INCR($LIBT006),$LIBEPN,$LIBT008,
         $LIBT009,$LIBT010,$LIBT011)
   ENUMV(INCR($LIBT011),$LIBDLL)
```

$LNAMET TYPE $LNAMET

Library Member Name

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $LBDEL $LBFIND $LBGTLSN $MSGCXO
USED in TYPE....: $LIBENTRY $MEMBER $RELPID
```

```
TYPDEF $LNAMET [8] CHAR;
```

LONG TYPE LONG

DASL scalar data type: 4 bytes signed

```
ENTERED.........: 83 Apr 20    UPDATED.:
  *** DASL COMPILER DEFINED, primary data TYPE ***

      LSW.LSB LSW.MSB MSW.LSB MSW.MSB ?
```

$LSN TYPE $LSN

Logical Sector Number

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED in TYPE....: $ACB  4 times  $FILEINFO 2 times
                  $ICB  2 times  $FILEPTR
                  $OPENPT        $PFDB     $SECURETBL

TYPDEF $LSN ULONG;
```

$MAB TYPE $MAB

Managed File Access Block

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$FAR
USED in TYPE....: $DCB (managed)

TYPDEF $MAB STRUCT {
  $MABID    UNSIGNED; ID of entity being managed
  $MABLINK  STRUCT {        Links
            $LNK1  UNSIGNED;
            $LNK2  UNSIGNED; };
  $MABUAT   UNSIGNED; User access token (UAT) ID
  $MABLGON  UNSIGNED; Log-on pipe FAV
  $MABRQIN  UNSIGNED; Request init pipe FAV
  $MABPRIV  UNSIGNED; Private pipe
                      (request+response)FAV
  };
```

Library Member Structure

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 01
INCLUDE FILE....: D$UFRLIB
USED by FUNCTION: $LBADD $LBFIND

TYPDEF $MEMBER STRUCT {
   $LIBMTYP BYTE; Type of this member ($LIB...)
   $LIBMNAM $LNAMET; Member name
   $LIBMLSN UNSIGNED; First LSN
   $LIBMLEN UNSIGNED; Length in sectors
   $LIBMVRN BYTE; 3-bits version, 5-bits revision
   $LIBMLMD UNSIGNED; Last modification date
   };

NOTE: $MEMBER and $LIBENTRY are "identical"
      except $LIBMTYP is type BYTE in $MEMBER
      and type $LIBTYPE in $LIBENTRY
```

$NAMEEXT TYPE $NAMEEXT

Name, and Extension

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
POINTED TO in...: $OPENPT

TYPDEF $NAMEEXT STRUCT {
   $NAME $NAMET; Name
   $EXT  $EXTT; Extension
   };

NOTE: Blank fill with $WSBLANK character
      or normal blanks.
```

$NAMEEXTENV TYPE $NAMEEXTENV

Name, Extension, and Environment

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $AOPEN $GENSMSK $GENSTST
                  $IOPEN $IPREP   $NQDQLGN

TYPDEF $NAMEEXTENV STRUCT {
   $NAME $NAMET; Name
   $EXT  $EXTT; Extension
   $ENV  $ENVN; Environment
   };

NOTE: Blank fill with $WSBLANK character
      or normal blanks.
```

$NAMET TYPE $NAMET

File Name Type

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Jul 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $LOCKRIM $DONATFV $FSCAN
                  $SINDEP  $SLOCAL
USED in TYPE....: $ABSHDR   $FILEINFO     $INFOITEM    |*
                  $NAMEEXT  $NAMEEXTENV   $PIPEGENPT
                  $SFENT    $SFITABLE     $UABSECTOR
POINTED TO in...: $FILESPK

TYPDEF $NAMET [12] CHAR;

NOTE: Blank fill with $WSBLANK character
      or normal blanks.
```

$NODEFLAGS TYPE $NODEFLAGS

Node Definition Flags in $INFO

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSGEN
USED in TYPE....: $INFOITEM

```
TYPDEF $NODEFLAGS
       SET($INFOFF, $INFTXER, $INFCONG, $INFCFU,
           $INFIFS, $INFFMA);
```

$NQDQITEM TYPE $NQDQITEM

NQDQ List Item

ENTERED.........: 82 Jul 01   UPDATED.: 83 Feb 01
INCLUDE FILE....: D$UFRNQDQ
USED in TYPE....: $NQDQMSG array

```
TYPDEF $NQDQITEM [6] BYTE;
```

$NQDQMSG TYPE $NQDQMSG

NQDQ Message

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRNQDQ
USED by FUNCTION: $NQDQBLD $NQDQENL $NQDQENQ

```
TYPDEF $NQDQMSG STRUCT {
   $NQDQCNT  BYTE; Item count
   $NQDQLIST [0] $NQDQITEM; Item list
   };
```

NQDQ Statistics

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRNQDQ
USED by FUNCTION: $NQDQSTA

TYPDEF $NQDQSTAT STRUCT {      NQDQ Statistics
   CTNQACT  UNSIGNED; Number of active enqueues
   CTNQWAIT UNSIGNED; Nmbr of enqueues currently
                      waiting
   CTREQPND UNSIGNED; Number of requests pending
   CTFREECB UNSIGNED; Nmbr of free control blocks
                      (this is also the number of
                      additional users that could
                      logon)
   CTINACT  UNSIGNED; Number of inactive users
                      (users w/ no active enqueue
                      and no pending request)
   CTFRERLE UNSIGNED; Nmbr of free request list
                      elements
   CTUSERLE UNSIGNED; Nmbr of used request list
                      elements
   CTFREBUF UNSIGNED; Free buffers
   CTUSERS  UNSIGNED; Current number of logged on
                      users
   CTUSEBUF UNSIGNED; Number of used buffers
   CNTENQ       LONG; Number of enqueue requests
   CNTDEQ       LONG; Number of dequeue requests
   CNTLOGN      LONG; Number of logon requests
   CNTLOGF      LONG; Number of logoff requests
   CNTINV       LONG; Number of invalid requests
   CNTTO        LONG; Number of timed out enqueues
   };
```

Open Parameter Table

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMS
USED by FUNCTION: $AOPEN   $DOPEN  $ENVPEEL $ENVPLOP
                  $IOPEN   $IPREP  $NQDQBLD $OPEN
                  $OPENENV $RENENV $REOPEN  $TXOPEN
                  $TXOPENP $TXPREP $TXPREPP $WFOPEN
                  $WFOPENP $WFPREP $WFPREPP
USED in TYPE....: $OPENPTS
              I,...means field requires Input value
              O,...means field returns Output value
              B,...means Input required, Output returned

TYPDEF $OPENPT STRUCT {  Open Parameter Table
  $OTPFDB ^ $PFDB; I,Pointer to PFDB,
  $OTENV ^ CHAR; I-$OPENENV, B-$OPEN and others.       |
                 Pointer to user's ENV data area        |
                 (or entry in UET for $OPEN UFRs only)  |
    Five String Entry (each terminated with $ENVTERM):
                 1. Symbolic Network Name
                 2. Symbolic Node Name
                 3. Symbolic Resource Name
                 4. HSI
                 5. List of up to 20 Packed Passwords
    Same requirement as $ENVT.DATA  ($FILEKEY)          |

  $OTFILE ^ $NAMEEXT; I,Pointer to file name and ext
  $OTKIND     O,Resource Kind ( $DK...),   see Note 1

      ENUM($DKWS,$DKDISK,$DKPIPE,$DKPRINT,
           $DKCASS,$DKMAGT,$DKCOMM,$DKTIMER);
      ENUMV(8,$DKCARDR,$DKCARDP,$DKPTR,$DKPTP,
           $DK883M,$DK863M,$DKSMPLR,$DKRIM)
      ENUMV(16,$DKFAX)
```

..... continued

```
$OTSUBK BYTE; 0,Resource sub-kind( $SK..)see Note 1
$OTFEOFB BYTE; 0,End-of-File byte pointer
$OTFLEN $LSN; B,End-of-File location LSN,
$OTFINC UNSIGNED; B,Space increment in sectors,
$OTFMT                B,Format ( $FFMT...)
                      Must be set to create.

     ENUM($FFMTSYS,$FFMTTMP,$FFMTTXT,$FFMTISM,
         $FFMTL55,$FFMTRAC,$FFMTDBC);
     ENUMV( 7,$FFMTBAS,$FFMTMAC,$FFMTWPS,$FFMTJOB,
         $FFMTBIN,$FFMTUTX,$FFMTMFD)
     ENUMV(14,$FFMTUPF,$FFMTWPF,$FFMTAIM,$FFMTXFD)    |*
     ENUMV(20,$FFMTL66,$FFMTL80)
     DEFINE($FFMTRPM,32)
     ENUMV($FFMTRPM,$FFMTR55,$FFMTR66,$FFMTR80)
     ENUMV(40,$FFMTPTR)

$OTTIME $TIME; 0,Time of creation in binary
$OTSQL BYTE; B,Security level;0 to $SQLMAX ($SQL...)
$OTCODE BYTE; 0,Access code ($ACCODES)
              or file created flag
$OTONOS BYTE; 0,Optimum number of sectors to do; nbr |*
              of sectors in a track ($SECWO,$SECRO)  |*
$OTFID UNSIGNED; 0,File identification FDT-PCN
$OTRID BYTE; 0,Resource identifier ARV serial nbr
$OTNID UNSIGNED; ?
};
```

Note 1: Value returned by $OPENENV function if open is successful or if Error condition is other than $ECUMAV,$ECSI021, or $ECSI022.

Note : Refer to SPRM Vol.4 Sec. 2.2.1

Special $OPENENV in $OMCHECK,$OMREPAR Modes

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $OPENENV (in $OMCHECK, $OMREPAR Modes)

```
            I,...means function requires Input value
            O,...means function returns Output value

TYPDEF $OPENPTS UNION {
$O $OPENPT;
STRUCT {
   $OTX1 [9] BYTE;
   $OTHFDLSN BYTE; O,LSN of first HFD sector (9)
   $OTHFDLGT UNSIGNED; O,Nbr hash file directry sect
   $OTIDLSN BYTE; O,Disk LSN of identification sector
   };
STRUCT                    TAPES
   $OTX2 [8] BYTE;
   $OTRTRY BYTE; I,Maximum number of retries
   $OTBLKL UNSIGNED; O,Maximum block length
   };
};
```

$OPTION TYPE $OPTION

$SCANOS Option Specification

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Apr 12
INCLUDE FILE....: D$RMS
USED by FUNCTION: $SCANOS

TYPDEF $OPTION STRUCT {
   $OPTSON $SONT; Option field name
   $OPTFLG             Flags: Set by $SCANOS
                or $OPTFQOK initialized by code.
            SET($OPTFDEF,$OPTFVAL,$OPTFQOK);
   $OPTVAL BYTE; Option value length
                 Must be initialized to $OPTVCLR.
                 $SCANOS sets to $OPTVSET if the
                 option is found.
                 ENUMV(254, $OPTVSET, $OPTVCLR)
   $OPTMAX BYTE; Option value maximum length
                 (Initialize to 0 unless other
                  value is applicable)
   $OPTSTR [0] CHAR; Option value string.
   };           Should be GLOBAL variables
       for future to guarantee consecutive memory.

PROGRAM EXAMPLE:
  Use of option scanner:
    One or more $OPTIONs must be followed by $OPTTAIL.

opt1 $OPTION   := { 'HELP    ', 0, $OPTVCLR, 0 };
opt2 $OPTION   := { 'CODE    ', 0, $OPTVCLR, 6 };
opt2s [6] CHAR := '      ';
optt $OPTTAIL  := { $OPTTERM, 0 };
  IF $SCANOS(&opt1) && D$CFLAG THEN $ERMSG();
```

$OPTTAIL TYPE $OPTTAIL

$SCANOS $OPTION List Terminator

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 01
INCLUDE FILE....: D$RMS
USED in TYPE....: (follows 'n' $OPTION's)

TYPDEF $OPTTAIL STRUCT {
   $$OPTTERM BYTE; $OPTTERM is defined 0377
   $OPTTOT BYTE; Option global total count
   };
```

$PABENTRY TYPE $PABENTRY

Relocatable Program ID Sector PAB Entry

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $RELPID

TYPDEF $PABENTRY STRUCT {
                                    Program address block entry
      $PABTBFLG $PABFLAGS; PAB Flags ( $PAB...)
      $PABTBNAM [8] CHAR; PAB name
      $PABTBADR UNSIGNED; PAB address
      $PABTBLGT UNSIGNED; PAB length
      };
```

$PABFLAGS TYPE $PABFLAGS

Relocatable Program ID Sector PAB Flags

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $PABENTRY

TYPDEF $PABFLAGS
            SET($PABF000,$PABF001,$PABFDATA,$PABFCOMN,  |
                $PABFPS,$PABFTP,$PABFREL,$PABASS);

NOTE:  Refer to the LINK Utility in the RMS User's      |
       Guide.                                           |
```

Packed Password

ENTERED.........: 82 Jul 01 UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS UPDATED.: 83 Jun 09
USED by FUNCTION: $GETPASS $PAKPW $UNPAKPW
USED in TYPE....: $FILEKEY

```
TYPDEF $PACKPW [6] BYTE;
```

Passwords are stored in Environment Data after packing: each 8 character password is packed into 6 bytes in the following fashion: Ø4Ø is subtracted from each byte: the two high order bits are then discarded. The eight characters each now have six bits: these 48 bits are now taken as 6 bytes. EXAMPLE :

```
ASCII       C       O       D       E       W       O       R       D
OCTAL     Ø1Ø3    Ø117    Ø1Ø4    Ø1Ø5    Ø127    Ø117    Ø122    Ø1Ø4
 - Ø4Ø     Ø43     Ø57     Ø44     Ø45     Ø67     Ø57     Ø62     Ø44
ND Ø77     Ø43     Ø57     Ø44     Ø45     Ø67     Ø57     Ø62     Ø44
bin     1ØØØ11  1Ø1111  1ØØ1ØØ  1ØØ1Ø1  11Ø111  1Ø1111  11ØØ1Ø  1ØØ1ØØ
pack      1ØØØ111Ø  11111ØØ1  ØØ1ØØ1Ø1  11Ø1111Ø  111111ØØ  1Ø1ØØ1ØØ
octal       Ø216      Ø371      Ø45       Ø336      Ø374      Ø244
```

Constraint: the first two characters must not compress into all ones (Ø377) as this is interpreted as the end of available passwords. This seems to work out that any character except for _ (Ø137; underbar) is okay for the first character of password.

Program Control Region (externally defined)

ENTERED.........: 82 Jul 01   UPDATED.: 84 Jul 01
INCLUDE FILE....: D$PCR
with exceptions (*): D$RMS defines four locations.
UFRCDEFS/TXT defines six
locations not defined in DASL.

NOTE:
The PCR is the area at the logical beginning of memory allocated to each task operating in any RMS controlled processor. All locations are extenally defined by RMS so the DASL definition for each is of the form: **EXTERN** ***name type;***

Flag values are **DEFINED** in the DASL file D$PCR.

(The items which occur only in the RMS source file UFRCDEFS/TXT would also be defined as EXTERN if they had been implemented in DASL).

The word EXTERN has not been printed here due to space limits. Instead, the relative memory address in octal has been provided. The items have been arranged in order of memory location with the lowest location first.

The memory locations may in some cases change with RMS updates. Remember not to reference these locations by number, but by Location Name. (Flags should also be referenced by Name.)

(Some documentation puts the highest location first and refers to this as the TOP of the PCR.)

First of FIXED PCR Region and PCR Sector Definitions

```
000000 $PCRSECF [0]   BYTE; First byte of PCR sector
002000 $PCRMIND [0]   BYTE; Default bottom UET/CMD
                            stack address
007207 $PCRFOF [0] BYTE; First byte of fixed PCR area
 __"__ $PCRCIF2 UNSIGNED; 2nd Command Interpreter
                           Flags                        |*
  $PCRF2AW  00001 Task Running under Attached
                  Workstation Control
```

Cursor Positions for Control of Main Screen Window

```
007211 $PCRCWVL BYTE; Current Window Lower Vertical     |*
007212 $PCRCWVU BYTE; Current Window Upper Vertical
007213 $PCRCWHR BYTE; Current Window Right Horizontal
007214 $PCRCWHL BYTE; Current Window Left Horizontal

                      See values $WSLC, $WSRC, $WSBL
```

Command Interpreter State Control Area
Fields contain the LSNs of the Folowing Members

```
007215 $PCRCMDM $LNAMET;  Member Name being executed,
                          or spaces
007225 $PCRMLRC UNSIGNED; Check bits for Control Area
007227 $PCRFVUP UNSIGNED; Locked FAV for User Program
007231 $PCRFVCI UNSIGNED; Locked FAV for CMD INT
007233 $PCRSTTE UNSIGNED; Current Command Interpreter
                         State Flags (pointer to LSN)

007235 $PCRMMSG UNSIGNED; "MESSAGE" membr for CMD INT
007237 $PCRMLGF UNSIGNED;   "LOGOFF"
007241 $PCRMRNL UNSIGNED;   "RUNLINE"
007243 $PCRMNXL UNSIGNED;   "NEXTLINE"
007245 $PCRMMAL UNSIGNED;   "MAIL"
007247 $PCRMRET UNSIGNED;   "RETURN"
007251 $PCRMLGN UNSIGNED;   "LOGON"
```

..... continued

Batched Job Facility Control Information

```
007253 $PCRBJCM $LNAMET; BJF "WAIT" module membr name
007263 $PCRBJCN $NAMEEXT;BJF "WAIT" module name/ext
007303 $PCRBJCE $ENVN;   BJF "WAIT" module extension
```

Partition Memory Allocation Control Flags

```
007313 $PCRPSKA BYTE; Current number of PSKs allocatd
007314 $PCRPSKM BYTE; Maximum number of PSKs allowed
```

Command Interpreter Flags

```
007315 $PCRCMDF SETW(....); Command interpreter flags
  $PCRDFNH    00001 Inhibit signon/heading display
  $PCRDFEH    00002 Display all heading records
  $PCRDFSO    00004 CMD INT entered from signon prgm
  $PCRDFNW    00010 No workstation available
  $PCRDFFK    00020 Abort GETLINE$, KEYIN$ on func key
  $PCRDFCF    00040 Reset char font and translate table
  $PCRDFNC    00100 No STOP bar - clears HELP window    |*
  $PCRDFBJ    00400 Batched job facility is active
  $PCRDFWW    01000 Standard window was active
  $PCRDFWI    02000 Standard window is active
  $PCRDFMC    04000 Command Line was Menu-Generated     |*
  $PCRDFNS   010000 No STOP bar - does not clear        |*
                    window                              |*
  $PCRDFCW   020000 Current Window data valid
  $PCRDFML   040000 Menu line exists; $PCRCLEL points
                    to New Line
  $PCRDF2F  0100000 Secondary CI Flags Available
```

Printer Information

```
007317 $PCRCPFL BYTE;  Current printer form length
  $PTRFLDV      66  Printer Form Length DEFAULT Value
```

Chaining Information

```
007320 $PCRCHNF SET(...);   Chaining flags
  $PCRCFAC   000001  Chaining active
  $PCRCFCR   000002  Next command line ready
  $PCRCFFO   000004  Chain file open
  $PCRCFNA   000010  "No Abort" on $ERROR exit
  $PCRCFRS   000020  CHAIN File has been restarted

007321 $PCRCHNV [3] BYTE;    Address of entry vector;
                             CHAIN routine
007324 $PCRCHND ^ $NAMEEXTENV; Pointer to CHAIN CMD
                               filename/ext:env
```

Logging Information

```
007326 $PCRLOGF SET(....);  Logging flag
  $PCRLFAC   000001  Logging active
  $PCRLFSP   000002  Logging suspended
  $PCRLFEO   000004  Log only error messages
  $PCRLFNI   000010  Log note display inhibited
  $PCRLFFO   000020  Log file open
  $PCRLFHR   000040  HD/RU before each logged message

007327 $PCRLOGV [3] BYTE;   Address of entry vector;
                            LOG routine
007332 $PCRLOGD ^ $NAMEEXTENV; Pointer to LOG CMD
                               Filename/Ext:Env
```

...... continued

File-In-Error Informaton Returned by $GETSFI

```
007334 $PCRFIEI [0]   BYTE; $SFITABLE  Start of area
                    Note: Following not in DASL
 __"__ $PCREHSI              File HSI      *UFRCDEFS

007374 $PCRENAM  File name                 *UFRCDEFS
007410 $PCREEXT  File extension            *UFRCDEFS
007414 $PCRERES  Resource containing file  *UFRCDEFS
007430 $PCRENET  Net containing node       *UFRCDEFS
007444 $PCRENOD  Node containing resource  *UFRCDEFS
```

ERROR Information

```
007460 $PCRERRF SET(...);   Command Interpreter
                            $ERROR reason flags
  $PCREFBC    000001  $ERRC (BC reg) error code exists
  $PCREFRF    000002  Recursion flag
  $PCREFFF    000004  File in error flag
  $PCREFNN    000010  Net/node info present
  $PCREFTS    000020  Error Message is on top of
                      Command Stack
  $PCREFCI    000040  Error Message is in CI Message
                      Member

007461 $PCRCLEL ^     CHAR; Command line syntax error
                            location
007463 $PCRSTAT [38]  BYTE; State storage area

007531 $PCRABTF SET(...);   Abort reason flgs * D$RMS
  $PCRAFGA    000001  General abort          * D$RMS
```

COMMAND File NAME/EXT:ENV

```
007532 $PCRCMDN $NAMEEXT;   Filename/Ext of
                            command executing * D$RMS
007552 $PCRCMDE $ENVN;      Environment name of
                            command executing * D$RMS
```

KEYIN Translate Table

```
007562 $PCRKXT  [128] CHAR; Keyboard translate table;
                            this workstation
```

"YES" and "NO" First Letters

```
007762 $PCRNO   CHAR;     DEFINE($NO,  'N')
007763 $PCRYES  CHAR;     DEFINE($YES, 'Y')
```

PCR Region Pointers

```
007764 $PCR1STP ^ BYTE;   Address of first byte of
                          PCR fixed data area
 __"__ $CLOTOPP ^ BYTE;   Above top of CHAIN/LOG
                          overlays

007766 $CLO1STP ^ BYTE;   Address of first byte in
                          CHAIN/LOG overlays
 __"__ $UETTOPP ^ BYTE;   Above top of UET area

007770 $UETPTR  ^ $ENVT;  Address of first UET entry
                          (linked list head)
007772 $CSTOPP  ^ BYTE;   Above top of command stack
 __"__ $UET1STP ^ BYTE;   Address of first byte in
                          user ENV table
007774 $CS1STP  ^ BYTE;   Address of first byte in
                          command stack
 __"__ $ULSTOPP ^ BYTE;   Above top of user logical
                          space
007776 $PCRMINP ^ BYTE;   Minimum usable PCR location

010000 $PCRTOP  [0] BYTE; First byte after top of PCR
 __"__ $ULS1ST  [0] BYTE; First byte of user logical
                          address space
```

....this is where you may put your program.

Physical File Descriptor Block

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Apr 12
INCLUDE FILE....: D$RMS
USED by FUNCTION: $CLOSE   $DONATFV $$FILEPCU $FORMAT
                  $LBADD   $LBDEL    $LBFIND  $LBFRES
                  $LBGTLSN $LOAD     $LOCKFAV $LOCKRIM
                  $MAP4K   $MSGCGET  $PIPEUSE $RUN
                  $$RUN    $SCANCFG  $SECCHK  $SECEOF
                  $SECR    $SECRO    $SECW    $SECWAIT
                  $SECWO   $SINDEP   $STOPIO
                  $TIMER   $UNLKRIM
USED in TYPE....: $ACB 2 times $DCB     $ICB    $FCBDIR
                  $FCBDIRPRT   $FCBDOVR $FCBPRT
POINTED TO in...: $DCB 2 times  $OPENPT
                  $RLPARAM      $WFCB 2 times
```

```
TYPDEF $PFDB STRUCT {
   $PFVID UNSIGNED; 16-bit FAV Identification
                     Returned by OPEN function
                    (Undefined for WORKSTATION)
                    (Only field required for TIMER)
   UNION {
     $PCLSN $LSN; Logical Sector Number
          DISK: Set to first sector to read or write.
      STRUCT {
         UNION {
            $PBLKL UNSIGNED; Blk lgth;bytes:PIPE,TAPE
                   Read: returns number of bytes read.
                   If $SECERR bit set, read incomplete.
                   Write: Set to number to be written.
                  (Returns number of bytes transfered.)
            STRUCT {
               $PSYSCODE BYTE; DIRECT RIM system code
               $PTASKID BYTE; DIRECT RIM task ID
               };
            };
```

..... continued

```
        UNION {
           $PTIMER BYTE; Timeout count in seconds:
                         Actual timeout varies -0 +1 secs.
                         $FOREVER waits until write occurs.
                         PIPE,DIRECT RIM
           $PSUBF BYTE; Sub-functon code: TAPE
                         Read Codes: $PTFREAD,$PTFRDRV,
           };              $PTFFSPB,$PTFBSPB,$PTFFSPF,
                           $PTFBSPF,$PTFRWND,$PTFULOD
        };               Write Codes: $PTFWRIT,$PTFWRTM,
                           $PTFERAS
     };                  7-Track: $PSUFCNV used with
                           $PTFREAD,$PTFRDRV,and $PTFWRIT
                           to un-block or block data.

   $PTODO BYTE; Number of 256-byte sectors to do
                PIPE: Must be >= $PBLKL / 256
   $PDONE BYTE; Number of 256-byte sectors done
   $PMXBF BYTE; Maximum number of buffers on list
   $PBUFL [1] $PFDBBUF; Start of buffer list
                    Must list buffers whose total
                    size is greater than $PTODO.
                    See D$BUFn's and $MAP4K.
```

$PFDBBUF TYPE $PFDBBUF

PFDB Buffer List Entry

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED in TYPE....: $ACB 2 arrays $DCB 1 array
$ICB 1 array $FCBDIR $FCBDIRPRT
$FCBDOVR $FCBPRT $PFDB

```
TYPDEF $PFDBBUF STRUCT {
   PSK  BYTE; Physical Mode:PSK
              Logical Mode: $NOPSK (or -1)
   PAGE BYTE; Physical Mode: Page number (0-15)
              Logical Mode: Page (0-255) in user's
                            logical address space
   SYS  UNSIGNED; System driver usage
   };
```

$PIPEGENPT TYPE $PIPEGENPT

Pipe Generation Parameter Table

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $PIPEGEN

```
TYPDEF $PIPEGENPT STRUCT {
   $PIPENAM  $NAMET; Name of Pipe
   $PIPEIAC  $ACCODES; Initial access code
   $PIPEKEY  $FILEKEYS; Keys
   $PIPETERM BYTE; Key list terminator:
   };              ($ENVTERM)
```

$RELCODE TYPE $RELCODE

Relocatable Sector Type Codes

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELCODE BYTE;
           ENUMV(0201,$RCPID,$RCOBJ,$RCEXDEF, $RCEXREF,
                      $RCXFER,$RCEPN,$RCLINEN)
```

$RELEPN TYPE $RELEPN

Reclocatable Entry Point Member Entry

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $RELEPNS

```
TYPDEF $RELEPN UNION {
      $RELEPNMBR STRUCT {          New program entry
             $RELEPNMFLG CHAR; New program flag
             $RELEPNMSKP [5] BYTE;
             $RELEPNMLSN UNSIGNED; Program LSN
             };
      $RELEPNAME [8] CHAR; Normal entry point name
      };
```

$RELEPNS TYPE $RELEPNS

Relocatable Entry Point Member Sector

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELEPNS [31] $RELEPN;
```

$RELLINE TYPE $RELLINE

Relocatable DEBUG Line Numbers Sector

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELLINE [84] $RELLNENT;
```

$RELLNENT TYPE $RELLNENT

Relocatable DEBUG Line Number Entry

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $RELLINE

```
TYPDEF $RELLNENT STRUCT {
      $RLNEPAB BYTE; Line number PAB
      $RLNEOFFS UNSIGNED; Line number offset
      };
```

$RELOBJ TYPE $RELOBJ

Relocatable Object Code Sector

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELOBJ [255] BYTE;
```

Relocatable Program ID Sector

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR
```

```
TYPDEF $RELPID STRUCT {
      $RPDXDPTR UNSIGNED; External definition pointer
      $RPDPGMNAM $LNAMET; Program name
      $RPDPABTABLE [16] $PABENTRY; PAB table
      };
```

Relocatable External Definition Sector

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR
```

```
TYPDEF $RELXDEF [22] $RELXDENT;
```

Relocatable External Definition Entry

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $RELXDEF
```

```
TYPDEF $RELXDENT STRUCT {
      $RXDENAME [8] CHAR; name
      $RXDEPAB  BYTE; PAB
      $RXDEVAL  UNSIGNED; value
      };
```

$RELXFER TYPE $RELXFER

Relocatable Starting Address Sector

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELXFER STRUCT {
      $RELXFERPAB BYTE; PAB
      $RELXFEROFFS UNSIGNED; Offset
      };
```

$RELXREF TYPE $RELXREF

Relocatable External Reference Entry

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $LIBSECTOR

```
TYPDEF $RELXREF [31] $RELXRENT;
```

$RELXRENT TYPE $RELXRENT

Relocatable External Reference Entry

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSSTRUCT
USED in TYPE....: $RELXREF

```
TYPDEF $RELXRENT UNION {
      $RELXREXNAM [8] CHAR; name
      $RELXRFWD STRUCT {   Forward reference definitn
             $RELFWDFLAG BYTE; forward reference flag
             $RELFWDPAB  BYTE; PAB
             $RELFWDOFFS UNSIGNED; Offset
             $RELFWDSKIP [4] BYTE;
             };
      };
```

Relocatable Loader Definition Structure

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
POINTED TO in...: $RLPARAM 2 times

```
TYPDEF $RLDEF STRUCT {
   $RLDNAME $RLNAME; Name
   $RLDVAL  UNSIGNED; Value
   };
```

Relocatable Loader Flags

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
USED in TYPE....: $RLPARAM

```
TYPDEF $RLFLAGS
                 SET($RLFNSVD,$RLFTOPD,$RLFEXDO,
                    $RLFRFUL,$RLFUNDF);
```

$RLNAME TYPE $RLNAME

Relocatable Loader Name Type

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$UFRRLD
USED by FUNCTION: $LOADREL
USED in TYPE....: $RLDEF

```
TYPDEF $RLNAME [8] CHAR;
```

$RLPARAM TYPE $RLPARAM

Relocatable Loader Parameters

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
USED by FUNCTION: $LOADREL

```
TYPDEF $RLPARAM STRUCT {
   $RLLSN   UNSIGNED; Member logical sector number
   $RLADDR  ^ BYTE; Starting load address
   $RLLIM   ^ BYTE; Limiting load address
   $RLPFDB  ^ $PFDB; Physical file descriptor blk adr
   $RLDEFAD ^ $RLDEF; User definition table address
   $RLDLIM  ^ $RLDEF; User definition table limit adr
   $RLREFAD ^ $RLREF; Usr reference work area address
   $RLRLIM  ^ $RLREF; Limit adr of reference wrk area
   $RLUDRTN ^ $RLUDRTNF; User defined symbol routine
   $RLUMEMA ^ $RLUMEMAF; User mem allocation routine
   $RLFLAG  $RLFLAGS; Load control flag
   };
```

$RLREF TYPE $RLREF

Relocatable Loader Reference Work Area

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
POINTED TO in...: $RLPARAM 2 times

```
TYPDEF $RLREF UNSIGNED;
```

$RLUDRTNF TYPE $RLUDRTNF

Relocatable Loader User Routine Types

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
POINTED TO in...: $RLPARAM

```
TYPDEF $RLUDRTNF ();
```

$RLUMEMAF TYPE $RLUMEMAF

Relocatable Loader User Routine Types

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRRLD
POINTED TO in...: $RLPARAM

```
TYPDEF  $RLUMEMAF ();
```

$RSRCFLAGS TYPE $RSRCFLAGS

$INFO Resource Status Flags

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 23
INCLUDE FILE....: D$RMSGEN
USED in TYPE....: $INFOITEM

```
TYPDEF $RSRCFLAGS
          SET($IRFOFF, $IRFOCP, $IRFWRP, $IRFCHK,
               $IRFSTP, $IRFSPC, $IRF6, $IRFBSD);
```

$SCFMSCODES TYPE $SCFMSCODES

FAR Exception Exit Codes

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$FAR

```
TYPDEF $SCFMSCODES
         ENUM($XCFMS00,$XCFMS01,$XCFMS02,$XCFMS03,
              $XCFMS04,$XCFMS05,$XCFMS06,$XCFMS07);
```

Exception conditions

$XCFMS01	File position out of range
$XCFMS02	No such record
$XCFMS03	ISAM key not found
$XCFMS04	ISAM duplicate key
$XCFMS05	Record already exists
$XCFMS06	No current record exists
$XCFMS07	File positioned to EOF

$SECSTAT TYPE $SECSTAT

Block I/O Status Control

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $SECCHK $SECWAIT |*

```
TYPDEF $SECSTAT
             SET($SECWP, $SECACT, $SECERR, $SECTMD,
                $SECFLOK, $SECSTOP, $SECSS, $SECBSD);
```

$SECURETBL TYPE $SECURETBL

$SECURE Parameter Table

ENTERED.........: 82 Jul 01   UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSIO
USED by FUNCTION: $SECURE

```
TYPDEF $SECURETBL STRUCT {
   $PFVID UNSIGNED; 16-bit FAV identifier;
                    initialized from PFDB                |*
   $SSFINC UNSIGNED; File increment
   UNION {
      STRUCT {                 $SSGET, $SSPUT modes
         $SSFSL BYTE; File security level
                      0 to $SQLMAX ($SQL...)
         $SSFIAC $ACCODES; File initial access code
         $SSKEYS $FILEKEYS; File access keys and codes |*
         $SSX [2] BYTE;
         };
      STRUCT {                 $SSGETX, $SSPUTX modes
         $SSFFMT BYTE; File format code ($FFMT...)
         $SSFCT $TIME; File creation time
         $SSSEGM BYTE; Number of segments in file
         $SSALLO $LSN; LSN of last allocated sector
         };
      };
   };
```

$SETIME Parameter Table

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMSGEN
USED by FUNCTION: $SETIME

TYPDEF $SETTIMEP STRUCT {
   $SETDIR BYTE; Clock adjustment direction
                           (2=up, 0=down)
   $SETADJ [3] BYTE; Clock adjustment in secds pr day
   $SETSEC $TIME; Time in secds since begning of 1901
   };
```

Symbolic File Table

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $SCANFS
USED in TYPE....: $FILESPK

TYPDEF $SFENT STRUCT {
   $SFTSFN $SFNT; Symbolic field name
                  (like 'IN' 'OUT' 'PRT' etc)
   $SFTNAM $NAMET; File name
   $SFTEXT $EXTT; File extension
   $SFTENV $ENVN; File environment
   };
```

$SFITABLE TYPE $SFITABLE

$GETSFI Information

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 01
INCLUDE FILE....: D$RMSIO
POINTED TO in...: $FILESTBL
USED in TYPE....: $PCRFIEI
```

```
TYPDEF $SFITABLE STRUCT {
   $GSFIHSI $HSI; File HSI
   $GSFINAM $NAMET; File name
   $GSFIEXT $EXTT; File extension
   $GSFIRES $NAMET; Resource containing the file
   $GSFINET $NAMET; Network containing the node
   $GSFINOD $NAMET; Node containg the resource
   };
```

$SFNT TYPE $SFNT

Symbolic File Name

```
ENTERED.........: 82 Dec 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMS
USED in TYPE....: $SFENT
```

```
TYPDEF $SFNT   [8] CHAR; like 'IN', 'OUT', 'PRT', etc. |*
```

$SONT TYPE $SONT

Symbolic Option Field Name

```
ENTERED.........: 82 Dec 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMS
USED in TYPE....: $OPTION
```

```
TYPDEF $SONT   [8] CHAR; like 'PRINT', 'INFO', 'SORT', |*
                     etc.                              |*
```

$STARTADR TYPE $STARTADR

Starting Address Type

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $LOAD $LOADREL $SLOCAL $TRAPUMV
USED in TYPE....: $INTS
```

```
TYPDEF $STARTADR ^ D$CALLF;
```

$SYSTIME TYPE $SYSTIME

System Time

```
ENTERED.........: 82 Jul 01    UPDATED.: 83 Jul 29
INCLUDE FILE....: D$RMSGEN
USED by FUNCTION: $GETIME
```

```
TYPDEF $SYSTIME STRUCT {
   SECONDS $TIME; Seconds since beginning of 1901
               (MSB first, LSB last)
   MILLISECONDS BYTE; Eight millisecond counter:
   };        values between (256-125) and (256-1)
```

$SYSTINFO TYPE $SYSTINFO

System Time Information

```
ENTERED.........: 83 Jul 23    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$RMSGEN
USED by FUNCTION: $GETIME (in alt. S.C.)                    |*
                  $SETIME (in alt. S.C.)                    |*

TYPDEF $SYSTINFO STRUCT {
   $TMUTC    $TIME;  Universal Co-ordinated Time
   $TMTZ     $TIME;  Time Zone Offset from UTC
   $TMADJDR  BYTE;   Clock Adjustment Direction
             DEFINE($TMADJDN,0)Adjust Clock Down
             DEFINE($TMADJUP,2)Adjust Clock Up
   $TMADJ    [3] BYTE;   Clock Adj'ment in Seconds/Day    |*
   $TMDSTOS  BYTE;    DST Adjustment in Hours
   $TMDTSTR  $DSTINFO; DST Start Table
   $TMDTEND  $DSTINFO; DST End Table
   $TM8MS    BYTE;     Eight Millisecond Counter
   };
```

$TIME TYPE $TIME

Time in Seconds Since Beginning of 1901

```
ENTERED.........: 82 Jul 01    UPDATED.: 82 Dec 01
INCLUDE FILE....: D$RMS
USED by FUNCTION: $CVSTIME $DATETIM
USED in TYPE....: $INFOITEM $OPENPT $SECURETBL
                  $SYSTIME $SYSTINFO $UABSECTOR

TYPDEF $TIME [5] BYTE;
```

ULONG TYPE ULONG

24 Bit Number Structure

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Aug 08
INCLUDE FILE....: D$INC
USED in TYPE....: $INFOITEM $LSN $SYSTINFO D$GET24            |*

TYPDEF ULONG  STRUCT {
   LSW UNSIGNED;
   MSB BYTE;
   };
```

User Abend Header Sector Format

ENTERED.........: 83 Apr 02   UPDATED.: 83 Jul 23
INCLUDE FILE....: D$RMSSTRUCT

```
TYPDEF $UABSECTOR UNION {
   STRUCT {          LSN 0
      $UABNAME $NAMET; User task name
      $UABTIME $TIME; Time of abort
      $UABERR  BYTE; Error control
      $UABPROC BYTE; Processor type
      $UABNVER [5] BYTE; Nucleus version
      $UABSTKN BYTE; Nbr of level in user stack
      $UABSTBN BYTE; Nbr of entries in usr sectr tble
      $UABSTAT SET User state
               ($UABDUAL,$UABPWSX,$UABPWSA,
                $UABSHAR,$UABLCL);
      $UABKKSS UNSIGNED; Keyboard key seq trap addr
      $UABDKSS UNSIGNED; Display key seq trap adddr
      $UABFKS  UNSIGNED; Function key trap addr
      $UABUMVS UNSIGNED; User mode violatn trap addr
      $UABAKSS UNSIGNED; Abort key seq trap addr
      $UABLKSS UNSIGNED; Logoff key seq trap addr
      $UABMMIN BYTE; Minimum memory allocation
      $UABMCUR BYTE; Current memory allocation
      $UABMMAX BYTE; Maximum memory allocation
      $UABPCRK BYTE; PCR Sector PSK
      $UABPWSK BYTE; PWS Sector PSK
      $UABDAD  $NAMET; Father task name
      $UABSHID $NAMET; Shared program name
      $UABFAVA BYTE; Nbr of active I/O operations
      $UABFAVC BYTE; Nbr of complete I/O operations
      $UABREG  [8] BYTE; User registers ABCDEHLX
      $UABFLG  BYTE; User flags
      $UABBR   BYTE; User base register
      $UABSTK  [32] UNSIGNED; User stack
      $UABSTB  [32] BYTE; User sector table
      };
```

..... continued

```
   $UABPSKS [128] STRUCT {  LSN 1 & 2
      $UABPSK BYTE; User PSK
      $UABSPSK SET Status of user PSK
            ($UABPRO,$UABPPRV,$UABPSHR,
             $UABPDAD,$UABPPCR,$UABPPWS);
      };
   $UABDATA [256] BYTE; LSN 3 thru n
   };
```

USER ABEND memory allocation table format
(LSN 1 & 2)

The table consists of 1 entry for each PSK allocated to the user. Each entry consists of a PSK byte followed by a status byte as defined. The table is terminated by a PSK byte of $NOPSK. Two sectors are always allocated for the memory allocation table to allow for the maximum user allocation of 1020k.

USER ABEND memory dump (starting in LSN 3)

The USER ABEND memory dump consists of 16 disks sectors per PSK allocated to the user. The memory dump order is defined by the memory allocation table in LSNs 1 and 2.

$UNPACKPW TYPE $UNPACKPW

Unpacked Password

ENTERED.........: 82 Jul 01   UPDATED.: 82 Dec 01
INCLUDE FILE....: D$UFRENV
USED by FUNCTION: $PAKPW $UNPACKPW

```
TYPDEF $UNPACKPW [8] CHAR;
```

UNSIGNED TYPE UNSIGNED

DASL scalar data type: 2 bytes unsigned

ENTERED.........: 83 Apr 20   UPDATED.:
*** DASL COMPILER DEFINED, primary data TYPE ***

$WFCB TYPE $WFCB

Work File I/O Control Block

```
ENTERED.........: 82 Jul 01    UPDATED.: 84 Jul 01
INCLUDE FILE....: D$UFRWFIO
USED by FUNCTION: $TAPEREWIND $TAPEUNLOAD
                  $TXBKSP  $TXCLOSE  $TXDEL   $TXOPEN
                  $TXOPENP $TXPOSEF  $TXPOSIT $TXPREP
                  $TXPREPP $TXREAD   $TXUPDAT $TXWEOF
                  $TXWRITB $TXWRITE  $WFCLOSE $WFFLUSH |*
                  $WFOPEN  $WFOPENP  $WFPOSEF $WFPOSIT |*
                  $WFPREP  $WFPREPP  $WFREAD  $WFREADL |*
                  $WFUPDATE $WFUPDATEL $WFWEOF
                  $WFWRITE $WFWRITEL
POINTED TO in...: $WFIOPRTN
```

```
TYPDEF $WFCB STRUCT {
   $WFCBRSIZ   UNSIGNED; Record size
   $WFCBFLAG   $WFCBFLAG; Control flag ( $WFF...)
   $WFCBPFDBP  ^ $PFDB; $PFDB pointer
   UNION {
         $WFCBCURR    $FILEPTR; current pos pointer
         $WFCBBLKCURR UNSIGNED; current block pos
                                non-disk
      };
   UNION {
         $WFCBEOF     $FILEPTR; EOF pointer
         $WFCBBLKSIZE UNSIGNED; current block size
                                non-disk
      };
```

.....continued

```
   UNION {
         $WFCBMAX    $FILEPTR; disk file maximum pntr
         STRUCT {
            $WFCBBLKMAX   UNSIGNED; max block size
                                    allowed
            $WFCBBLKCOUNT UNSIGNED; block I/O counter
            };
      };
   UNION {
         $WFCBPIO    ^ BYTE; Physical I/O routine
                             table address
         $WFCBPIOTAB ^ $WFIOTABPRTN; Routines
                       ordered: read, write, EOF set,
         };            EOF get, seek, close, open.
         $WFCBPFDBP2 ^ $PFDB; Secondary (double
                              buffering) PFDB
         $WFCBHOLD   BYTE; space compress'n hold area
         $WFCBFLAG2  $WFCBFLAG2; second control flag
                                 byte
         $WFCBRESV   [4] BYTE; Reserved Area
         };
```

$WFCB is generally initialized by the UFR.
Sometimes a $WFF.. format must be specified.

Physical Work File I/O Driver Routines

```
EXTERN WFPIO$, DBLBUFF$, PRTIO$, PRTDIO$,
       PIPEIO$, PIPEDIO$, PIPTIMO$ BYTE;
EXTERN TAPEDIO$, TAPEIO$ BYTE;
EXTERN VIRTUAL$, VBASEIO$ BYTE; Is VBASEIO$ a routine ?

EXTERN VIOINIT$ BOOLEAN; something for VIRTUAL ?
```

$WFCBFLAG TYPE $WFCBFLAG

Work File I/O Flags

ENTERED.........: 82 Jul 01   UPDATED.: 83 Apr 12
INCLUDE FILE....: D$UFRWFIO
USED in TYPE....: $WFCB

```
TYPDEF $WFCBFLAG
              SET( $WFFUPDEOF, $WFFDIRTY, $WFFSPCTXT,
                   $WFFUNCOMP, $WFFSHRD, $WFFNOTDSK,
                   $WFFINPROG, $WFFEOFOK );
```

$WFCBFLAG2 TYPE $WFCBFLAG2

Second Control Flag Byte Type

ENTERED.........: 83 Apr 02   UPDATED.:
INCLUDE FILE....: D$UFRWFIO
USED in TYPE....: $WFCB

```
TYPDEF $WFCBFLAG2 SET($WFPACKED);
```

$WFIOPRTN TYPE $WFIOPRTN

Physical I/O Function Routine Type

ENTERED.........: 83 Apr 02   UPDATED.:
INCLUDE FILE....: D$UFRWFIO
USED in TYPE....: $WFIOTABPRTN

```
TYPDEF $WFIOPRTN  (wfcb ^ $WFCB) D$CCODE;
```

$WFIOTABPRTN TYPE $WFIOTABPRTN

Table of Pointers to Phys I/O Routines

ENTERED.........: 83 Apr 02   UPDATED.: 83 Jul 30
INCLUDE FILE....: D$UFRWFIO
POINTED TO in...: $WFCB

```
TYPDEF $WFIOTABPRTN  [7] ^ $WFIOPRTN;
```

$WCONFIG Status Bits

NTERED.........: 82 Jul 01   UPDATED.: 84 Jul 01
NCLUDE FILE....: D$RMSWS
SED by FUNCTION: $WCONFIG

```
YPDEF $WSCONF
        SETW($WS0,$WS1,$WS2,$WS3,$WSCIAKA,
             $WSCKDKA,$WSCIAUA,$WSCCKA);
        SETV(256,$WSBFSHA,$WSBFKUA,$WSBFKDA,$WSBFKSA,
                 $WSBLCFA,$WSBIVA,$WSBCPA,$WSBSWA)
it Definitions:
            Mask the following four bits with $WSKMASK |*
            and then compare to values $SKWS...:        |*
 $WS0           1 Workstation Kind bit0                 |*
 $WS1           2 Workstation Kind bit1                 |*
 $WS2           4 Workstation Kind bit2                 |*
 $WS3         010 Workstation Kind bit3                 |*

 $WSCIAKA     020 INT & ATT keys downstrokes avail.
 $WSCKDKA     040 KBD & DPY keys static bits &
                  upstrokes avail
 $WSCIAUA    0100 INTERRUPT & ATTENTION keys static
                  bits and upstrokes available
 $WSCCKA     0200 Click available
 $WSBFSHA    0400 Shiftd function keys available
 $WSBFKUA   01000 F1 thru F5 upstrokes avail
 $WSBFKDA   02000 F1 thru F5 downstrokes avail
 $WSBFKSA   04000 F1 thru F5 static bits avail
 $WSBLCFA  010000 Display font set loadable
 $WSBIVA   020000 Inverted video available
 $WSBCPA   040000 Cursor positioning avail
 $WSBSWA   010000 Sub windows available
```

$WCONFIG Third Status Byte Flags

ENTERED.........: 83 May 03   UPDATED.: 84 Jul 01
INCLUDE FILE....: D$RMSWS
USED in TYPE....: $WCONFDS

```
TYPDEF $WSCONF2 SET
                    ($WS2POW,$WS2IPL,$WS2CFL,
                     $WS22LVA,$WS2ULNA,$WS2BNKA,
                     $WS2EFKO);                         |*

Value Definitions:                                      |*
  $WS2POW     1  Tube just powered on
  $WS2IPL     2  System just Re-Booted
  $WS2CFL     4  Cursor font loadable
  $WS22LVA  010  2-Level video available
  $WS2ULNA  020  Underline available
  $WS2BNKA  040  Blink available
  $WS2EFKO 0100  Expanded function keyboard on-line    |*
```

$WCONFIG 4 Byte Status Structure

ENTERED.........: 83 May 03 UPDATED.: 84 Jul 01
INCLUDE FILE....: D$RMSWS
USED by FUNCTION: $WSIO (CTL Code $WSCONFD $WSRECON ?)

```
TYPDEF $WSCONFDS STRUCT {
   $WSCONC $WSCONF; First Two Bytes
   $WSCON2 $WSCONF2; Third Status Byte
   $WSCON3         Processor Dependent
                      (see RMSSYSxx/TEXT)
           ENUM($WS3FD0,$WS3FD1,$WS3FD2,$WS3FD3,
                $WS3FD4);                                      |*

      DEFINE($WS3FDMK,017)  'Font Data' Mask
      DEFINE($WS3NFMK,0360)  'Number of Fonts' Mask
      DEFINE($WS3NFS,4)  'Number of Fonts' Shift Value
   };
```

		Font Characteristic Indicators				
	Val	Size Horiz	Size Vert	Length Per Char	Descriptor Required?	
$WS3FD1	1,	5	7	5	no	
$WS3FD2	2,	5	7	7	no	
$WS3FD3	3,	8	12	12	yes	
$WS3FD4	4,	9	12	12	yes	|*
$WS3FD0	0,	Character Font Not Loadable				

$WSIO Mode Bits

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 23
INCLUDE FILE....: D$RMSWS
USED by FUNCTION: $WSIO

```
TYPDEF $WSIOMODE
        SET($WSMNW, $WSMES, $WSMNI, $WSMNE, $WSMKCON,
            $WSMDIGO, $WSMPADN, $WSMNESC);
Definitions:
  $WSMNW   0001  Inhibit 'DISPLAY' key wait
  $WSMES   0002  Echo secret (char)
  $WSMNI   0004  No case inversion
  $WSMNE   0010  No echo or cursor
  $WSMKCON 0020  Keyin continuous
  $WSMDIGO 0040  Digits only
  $WSMPADN 0100  Pad numeric decimal part
  $WSMNESC 0200  No escape before 0-037 or 0177
```

$WSTATUS Status Bits

ENTERED.........: 82 Jul 01   UPDATED.: 83 Jul 23
INCLUDE FILE....: D$RMSWS
USED by FUNCTION: $WSTATUS, $WAITOS

```
TYPDEF $WSTAT
                SETW($WSF1,$WSF2,$WSF3,$WSF4,$WSF5,
                     $WSDSP,$WSKBD,$WSONL);
                SETV(256,$WSRDY,$WSINT,$WSATT)
```

Definitions:

$WSF1	0001	'F1' key down
$WSF2	0002	'F2' key down
$WSF3	0004	'F3' key down
$WSF4	0010	'F4' key down
$WSF5	0020	'F5' key down
$WSDSP	0040	'DISPLAY' key down
$WSKBD	0100	'KEYBOARD' key down
$WSONL	0200	'ONLINE' status (always true)
$WSRDY	0400	Key ready
$WSINT	01000	INTERRUPT key down
$WSATT	02000	ATTENTION Key Dowm

CROSS REFERENCE SECTION

FORMAT

WordCode *WordName* *Value*(Octal if Leading 0, else Decimal)....
.... *Description*...... [: *TYPE ref*] : *FILE Ref*

WORD CODES

abr	Abbreviation (does not belong in program code)
cr	Cross Referanced Function
dcm	DASL Compiler Macro
dcw	DASL Control Word
ddt	DASL Data Type
ddw	DASL Declaration Word
def	DASL External Function 'lacro
dim	DASL Include File Macro
drw	DASL Reserved Word (not dcm,dcw,ddt,ddw)
far	File Access Routine Macro
fld	Field of a TYPEDEF'd structure
inc	DASL INCLUDE File Name
sc	System Call Routine Macro
typ	TYPDEF defined Structure
ufr	User Function Routine Macro
val	Value defined in one several ways: SET, SETV, SETW, ENUM, ENUMV, DEFINE
var	Variable declared with EXTERN statement

Where to find more info based on WORD CODE:

abr	No place specific, abreviations are just for help.
cr	FUNCTIONS Section, and the *System Prog. Ref. Manual*.
dcm	FUNCTIONS Section, and the *DASL Document*.
dcw	FUNCTIONS Section, and the *DASL Document*.
ddt	TYPES Section, and the *DASL Document*.
ddw	FUNCTIONS Section, and the *DASL Document*.
def	FUNCTIONS Section, and the *DASL Document*.
dim	FUNCTIONS Section, and the *DASL Document*.
drw	FUNCTIONS Section, and the *DASL Document*.
far	FUNCTIONS Section, and the *System Prog. Ref. Manual*.
fld	TYPES Section under TYPE specified at end of line.
inc	LISTS section has functions and types in each file.
sc	FUNCTIONS Section, and the *System Prog. Ref. Manual*.
typ	TYPES Section, shown as defined in INCLUDE File.
ufr	FUNCTIONS Section, and the *System Prog. Ref. Manual*.
val	Some shown in TYPES Section with TYPE defining them. Also see LIST's section of flags and values.
var	Some are referenced in TYPES or FUNCTIONS using them.

REFERENCE NAME at end of Word Line:

What the names after the colon mean.

CODES	
abr	None
cr	None
dcm	None
dcw	None
ddt	None
ddw	None
def	: **D$*something***, DASL INCLUDE file that defines DASL function interface to the externally defined function which is contained in the relocatable code library ***D$LIB/REL***.
dim	: **D$*something***, DASL INCLUDE file where the DASL Macro is defined.
drw	None
far	: **D$FAR** typically, the DASL INCLUDE file that defines the DASL function's interface to the externally defined RMS System function contained in relocatable code library ***FAR/REL***.
fld	: ***type*** : **D$*something***, the name of the DASL TYPE which contains the named variable field, and the DASL INCLUDE File that defines the TYPE.
inc	None
sc	: **D$*somthing***, the DASL INCLUDE file that defines the DASL function's interface to the RMS Nucleus System Routine function.
typ	: **D$*something***, DASL INCLUDE file where TYPE defined.
ufr	: **D$*something***, the DASL INCLUDE file that defines the DASL function's interface to the externally defined RMS System function contained in relocatable code library ***RMSUFRS/REL***.
val	: **D$*something***, DASL INCLUDE file where value defined. : ***type*** : **D$*something***, some values are defined within the definition of a TYPE, in which case that TYPE is listed as well as the DASL INCLUDE file.
var	: **D$*something***, DASL INCLUDE file where the externally defined variable is declared so that its *address* is LINKED into the program as the value of its *name*.

the WORDS

val	***A$ABORT***	AIMDEX Aborted (Invalid Index)	
	A$ABORT	0001	: $FCBA.$FCBFLAG2 : D$FAR
val	***A$PMISS***	Primary record key must mis-match	
	A$PMISS	0020	:$FCBA.$FCBFLAG2 : D$FAR
val	***A$PRISEL***	0040 Primary select active	:$FCBA.$FCBFLAG2 : D$FAR
val	***A$SHARE***	0200 Shared AIM index	:$FCBA.$FCBFLAG2 : D$FAR
val	***A$UPCASE***	0100 Force upper case	:$FCBA.$FCBFLAG2 : D$FAR
abr	***ABEND***	"Abort End Program, Dump Memory to File"	
abr	***ABS***	Absolute	
typ	***$ABSHDR***	STRUCT; Lib Absolute Element Hdr Sec	: D$RMSSTRUCT
fld	***$ABSSECTOR***	UNION; Absolute member sectors	: $LIBSECTOR
val	***$ACATALG***	010 Obtain catalog information	:$ACCODES:D$RMSIO
abr	***ACB***	AIM Control Block	
typ	***$ACB***	STRUCT, AIM control block	: D$FAR
fld	***$ACBACC***	BYTE; File access flags	: $ACB
fld	***$ACBBASE***	$LSN; Data file base LSN	: $ACB
fld	***$ACBBUFS***	BYTE; Nbr of buffers (todo)	: $ACB
fld	***$ACBCALL***	UNSIGNED; AIM user	: $ACB
fld	***$ACBCLSN***	$LSN; Current LSN	: $ACB
fld	***$ACBCNFG***	BYTE; Index config byte	: $ACB
fld	***$ACBDCUR***	$FILEPTR; Data file cursor	: $ACB
fld	***$ACBECNT***	UNSIGNED; Number of extension maps	: $ACB
fld	***$ACBELSN***	$LSN; LSN of extension index	: $ACB
fld	***$ACBETXT***	[4] BYTE; Start of data expansion	: $ACB
fld	***$ACBEXCL***	[7] BYTE; Map of excluded key fields	: $ACB
fld	***$ACBFALS***	UNSIGNED; False hits	: $ACB |*
fld	***$ACBFCUR***	BYTE; Free float buffer cursor	: $ACB
fld	***$ACBHUP***	CHAR; Highest upper-case character	: $ACB
fld	***$ACBIGNR***	CHAR; Don't care character	: $ACB
fld	***$ACBINDX***	$LSN; LSN of current index header	: $ACB
fld	***$ACBIOST***	BYTE; AIM I/O status byte	: $ACB
fld	***$ACBKEYL***	BYTE; Aggregate key length	: $ACB
fld	***$ACBKEYN***	BYTE; Number of keys configured	: $ACB
fld	***$ACBLUP***	CHAR; Lowest upper-case character	: $ACB
fld	***$ACBMCUR***	UNSIGNED; Access map cursor	: $ACB
fld	***$ACBMLSN***	$LSN; LSN of current AIM map	: $ACB
fld	***$ACBMSTR***	BYTE; Number of master keys	: $ACB
fld	***$ACBNMAP***	UNSIGNED; Map number	: $ACB
fld	***$ACBPCNT***	UNSIGNED; Nbr of primary maps	: $ACB
fld	***$ACBPFDB***	$PFDB; Start of the AIM index PFDB	: $ACB
fld	***$ACBPMLN***	BYTE; Length of primary maps	: $ACB
fld	***$ACBPSEG***	BYTE; Nbr of primary segments/sector	: $ACB
fld	***$ACBPSLN***	BYTE; Primary segment length	: $ACB
fld	***$ACBSKEY***	[8] BYTE; Pri recd select key	: $ACB
fld	***$ACBSLGT***	BYTE; Pri recd select key length	: $ACB
fld	***$ACBSLOC***	^ CHAR; Pri recd select key location	: $ACB
fld	***$ACBSTAT***	BYTE; AIM status byte	: $ACB
fld	***$ACBTMAP***	[32] BYTE; Current triplet map	: $ACB
fld	***$ACBWFDB***	$PFDB; Start of the work buffer PFDB	: $ACB
typ	***$ACCODES***	File Access Codes	: D$RMSIO
abr	***ACK***	Acknowledgement	
val	***$ACKILL***	0100 Delete the file	: $ACCODES : D$RMSIO

far	***$ACLOSE***	Close an AIM File : D$FAR
val	***$ACMAX***	000376 Sum of the $ACCODES : $ACCODES : D$RMSIO
val	***$ACREAD***	0002 Read data : $ACCODES : D$RMSIO
val	***$ACREATE***	0020 Create files under this catalog
	$ACREATE	: $ACCODES : D$RMSIO
val	***$ACRENM***	040 Rename the file : $ACCODES : D$RMSIO
val	***$ACREPX***	DEFINE($ACREPX,$ACSECQ) : $ACCODES : D$RMSIO
	$ACREPX	000200 Exclusive access,disk $OMREPAR only
val	***$ACSECQ***	0200 Change security info : $ACCODES : D$RMSIO
abr	***ACU***	Automatic Calling Unit
val	***$ACWRIT***	0004 Write data and deallocate space
	$ACWRIT	: $ACCODES : D$RMSIO
far	***$ADELCR***	AIM Delete Cur Rec Read by $AREAD,$AREADKG : D$FAR
abr	***AFN***	Associated File Number
abr	***AFV***	Available File Varible ?$ECFMT 2 No more AFVs
abr	***AIM***	Associative Indexed Method
far	***$AINS***	Ins Key in AIM Idx for Existing Data Recd : D$FAR
far	***$AIOCLR***	Complete Pend Writes, Buf Fill on Reads : D$FAR
val	***$ALTSC***	0200, Alternate SC mode select : D$INC
abr	***ANSI***	American National Standards Institute
abr	***ANV***	Available Node Variable
far	***$AOPEN***	Initialize AIM File Access : D$FAR
far	***$APOS***	Pos to Logical Rec Meeting Key List Spec : D$FAR
abr	***ARC***	Attached Resource Computing
far	***$AREAD***	AIM Read Logical Rec Meeting Key List Spec : D$FAR
far	***$AREADCR***	Read after $APOS/Re-read Cur after $AREAD : D$FAR
far	***$AREADKG***	AIM Read Key Generic, Read Recs w/ Same Key : D$FAR
abr	***ARV***	Available Resource Variable
far	***$ARWRTCR***	AIM Rewrite Cur Rec after $AREAD, $AREADKG : D$FAR
abr	***ASCII***	American Standard Code for Information Interchange
abr	***ASM***	Assembler (language)
far	***$AWRITE***	Write Recd at End of Data, Ins Key in Index : D$FAR
sc	***$BASESET***	Set the Memory Base Register : D$RMSMEM
sc	***$BEEP***	Workstation, Make a Beep Sound : D$RMSWS
val	***$BFTDATA***	000002 Data file buffer type : D$FAR
val	***$BFTINDX***	000001 ISAM index buffer type : D$FAR
val	***$BFTOTHR***	000003 Other buffer type : D$FAR
cr	***BIGBCP$***	Compare Large Strings; Assembly Language UFR
cr	***BIGBT$***	Move 16-bit Length; Assembly Language UFR
ufr	***$BINOC24***	Convert 24-Bit Binary to ASCII Octal : D$UFRNUM
ufr	***$BINOCT***	Convert 16-bit Binary to ASCII Octal : D$UFRNUM
abr	***BJF***	Batch Job Facility
ddt	***BOOLEAN***	DASL scalar data type: 1 byte unsigned
abr	***BUF***	Buffer
abr	***BUFR***	Buffer |*
ddt	***BYTE***	DASL scalar data type: 1 byte unsigned
dcw	***CASE***	Transfer Control to Statement Number = Argument
abr	***CAT***	Cluster Allocation Table
abr	***CCF***	Central Configuration File
abr	***CDF***	Constant Definitions File
abr	***CFD***	Character Font Database

fld	***$CFGADDR***	^ BYTE; Output address	: $CFGKEY
typ	***$CFGEND***	BYTE; $SCANCFC Keyword List Terminator	: D$UFRSCAN
fld	***$CFGFLAG***	BYTE; Control byte	: $CFGKEY
val	***$CFGFND***	0200 Keyword found : $CFGKEY.$CFGFLAG	: D$UFRSCAN
val	***$CFGGTN***	0040 Too many values : $CFGKEY.$CFGFLAG	: D$UFRSCAN
typ	***$CFGHDR***	[13] CHAR; $SCANCFG Configuration Header	:D$UFRSCAN
typ	***$CFGKEY***	STRUCT; $SCANCFG Keyword Parameters	: D$UFRSCAN
fld	***$CFGKEYW***	[9] CHAR; Keyword	: $CFGKEY
val	***$CFGLONG***	0020 Value too large : $CFGKEY.$CFGFLAG	:D$UFRSCAN
fld	***$CFGNOKS***	BYTE; Nuber of values	: $CFGKEY
val	***$CFGNOVM***	0017 (4 bits) Number of values moved	
	$CFGNOVM	: $CFGKEY.$CFGFLAG	:D$UFRSCAN
fld	***$CFGTYPE***	BYTE; Type flag	: $CFGKEY
fld	***$CFGVLGT***	BYTE; Value length	: $CFGKEY
val	***$CFGVPT***	0100 Value type error : $CFGKEY.$CFGFLAG	:D$UFRSCAN
abr	***CFT***	Character Font Table	
ufr	***$CHAININ***	Workstn-IF, Determine if CHAINing is Active	:D$UFRWS
abr	***CHAR***	Character	
ddt	***CHAR***	DASL scalar data type: 1 byte unsigned	
abr	***CHN***	Chain	
abr	***CI***	Command Interpreter	
val	***$CILOG0***	0 Error in SIGNON	: D$RMSPROG
val	***$CILOG1***	1 Can not be loaded	: D$RMSPROG
val	***$CILOG2***	2 Command Interpreter can not be loaded	: D$RMSPROG
val	***$CILOG3***	3 Error in Command Interpreter	: D$RMSPROG
val	***$CILOG4***	4 Not enough memory for SIGNON	: D$RMSPROG
val	***$CILOG5***	5 SIGNON aborted	: D$RMSPROG
val	***$CILOG6***	6 LOGOFF key sequence	: D$RMSPROG
val	***$CILOG7***	7 LOGOFF forced from program	: D$RMSPROG
val	***$CILOG8***	8 ** UNUSED **	: D$RMSPROG
val	***$CILOG9***	9 Error in LOGOFF program	: D$RMSPROG
val	***$CISABT***	002 'ABORT' key sequence exit	: D$RMSPROG
val	***$CISACC***	10 Memory access violation during LOAD	: D$RMSPROG
val	***$CISADR***	9 Insufficient logical address space	: D$RMSPROG
val	***$CISDUAL***	11 Dual Sector Tables not supported	: D$RMSPROG
val	***$CISERRR***	001 'ERROR' system call exit	: D$RMSPROG
val	***$CISFMT***	06 Format error in progrm to execute	: D$RMSPROG
val	***$CISHAR***	12 Shared program mis-match	: D$RMSPROG
val	***$CISMEM***	010 Not enough mem for prgm execute	: D$RMSPROG
val	***$CISNORM***	000 'NORMAL' system call exit	: D$RMSPROG
val	***$CISREAD***	07 Read I/O err in prgrm to execute	: D$RMSPROG
val	***$CISUMV***	000003 User mode violation exit	: D$RMSPROG
val	***$CISVABT***	5 VANTAGE 'Abort' Key Sequence Exit	: D$RMSPROG
val	***$CISWSER***	000004 Workstation error exit	: D$RMSPROG
val	***$CISYSTB***	13 Insufficient System Table space	: D$RMSPROG
val	***$CK1403***	010 1403 Tortilla flex disk :$INFOITEM	: D$RMSGEN
val	***$CK88D1***	04 8800 SMD/MMD disk IMOD :$INFOITEM	: D$RMSGEN
val	***$CK88MEM***	0005 8800 memory bank :$INFOITEM	: D$RMSGEN
val	***$CK9301***	0006 9301 20 MB Whizzie :$INFOITEM	: D$RMSGEN
val	***$CK9310***	0007 9310 10 MB Cynthia :$INFOITEM	: D$RMSGEN
val	***$CK9315***	9 9315 10 MB Cyclone : $INFOITEM.$ICKIND	:D$RMSGEN
val	***$CK9324***	10 Moses Controller $INFOITEM.$ICKIND	:D$RMSGEN
val	***$CK9350***	000000 9350 Cartridge disk :$INFOITEM	: D$RMSGEN
val	***$CK9370***	01 9370 25 MB Mass stg disk :$INFOITEM	: D$RMSGEN
val	***$CK9374***	02 9374 10 MB Mass stg disk :$INFOITEM	: D$RMSGEN

val	*$CK9390*	03 9390 67 MB Mass stg disk :$INFOITEM	: D$RMSGEN
sc	*$CLICK*	Workstation, make a click sound	: D$RMSWS
abr	*CLK*	Clock	
var	*$CLO1STP*	^ BYTE; Adr of first byte in CHAIN/LOG ovls	: D$PCR
abr	*CLOP*	Central Logon Pipe	
sc	*$CLOSE*	Close a File	: D$RMS
sc	*$CLOSEAL*	Close All Open Files	: D$RMSIO
ufr	*$$CLOSEAL*	Interface to $CLOSEAL System Call	: D$UFRWS
var	*$CLOTOPP*	^ BYTE; Above top of CHAIN/LOG overlays	: D$PCR
abr	*CLP*	Central Logon Pipe Controller	
abr	*CLV*	Connection Link Variable	
val	*$CMCHOP*	000002 Deallocate to EOF LSN	: D$RMS
abr	*CMD*	Command	
val	*$CMKILL*	003 Delete the file	: D$RMS
val	*$CMSIZE*	001 Deallocate to specified LSN	: D$RMS
val	*$CMUNCH*	000 No change in file size	: D$RMS
fld	*CNTDEQ*	LONG; Number of dequeue requests	: $NQDQSTAT
fld	*CNTENQ*	LONG; Number of enqueue requests	: $NQDQSTAT
fld	*CNTINV*	LONG; Number of invalid requests	: $NQDQSTAT
abr	*CNTL*	Control	
fld	*CNTLOGF*	LONG; Number of logoff requests	: $NQDQSTAT
fld	*CNTLOGN*	LONG; Number of logon requests	: $NQDQSTAT
fld	*CNTTO*	LONG; Number of timed out enqueues	: $NQDQSTAT
fld	*$CODE*	BYTE; Error number	: $ERRCODE
ufr	*$CONDEC*	Convert Decimal to Binary	: D$UFRNUM
ufr	*$CONOC24*	Convert ASCII Octal to 24-Bit Binary	: D$UFRNUM
ufr	*$CONOCT*	Convert ASCII Octal to 16-Bit Binary	: D$UFRNUM
abr	*CP*	Continuation Pointer(buffer adr of first data byte)	
val	*$CP*	0240 Cursor position follows (vert),(horz)	:D$RMSWS
abr	*CRC*	Cyclic Redundancy Check	
abr	*CREP*	Central Request Pipe	
var	*$CS1STP*	^ BYTE; Addr of first byte in command stack	: D$PCR
ufr	*$CSCPUSH*	Push Command Lines Below Current Pointer	: D$UFRWS
ufr	*$CSPOP*	Workstn-IF, Pop The Command Stack	: D$UFRWS
ufr	*$CSPUSH*	Push a Command Line Onto Command Stac	: D$UFRWS
ufr	*$CSPUSHN*	Push Command Lines Onto Command Stack	: D$UFRWS
var	*$CSTOPP*	^ BYTE; Above top of command stack ?	: D$PCR
fld	*CTFREBUF*	UNSIGNED; Free buffers	: $NQDQSTAT
fld	*CTFREECB*	UNSIGNED; Number of free control blocks	: $NQDQSTAT
fld	*CTFRERLE*	UNSIGNED; Nbr of free req list elements	: $NQDQSTAT
fld	*CTFRERSL*	is not in DASL structure	: $NQDQSTAT
fld	*CTINACT*	UNSIGNED; Number of inactive users	: $NQDQSTAT
fld	*CTNQACT*	UNSIGNED; Number of active enqueues	: $NQDQSTAT
fld	*CTNQWAIT*	UNSIGNED; Number of enqueues waiting	: $NQDQSTAT
abr	*CTOS*	Casette Tape Operating System	
fld	*CTREQPND*	UNSIGNED; Number of requests pending	: $NQDQSTAT
fld	*CTUSEBUF*	UNSIGNED; Number of used buffers	: $NQDQSTAT
fld	*CTUSERLE*	UNSIGNED; Nbr of used req list elements	: $NQDQSTAT
fld	*CTUSERS*	UNSIGNED; Number of logged on users	: $NQDQSTAT
fld	*CTUSERSL*	is not in DASL structure	: $NQDQSTAT
abr	*CTV*	Controller Table Variable	
sc	*$CURSOFF*	Workstation, Turn Off the Cursor	: D$RMSWS
sc	*$CURSON*	Workstation, Turn On the Cursor	: D$RMSWS
ufr	*$CVB*	Convert ASCII Decimal to 16-Bit Binary	: D$UFRNUM
ufr	*$CVB24*	Convert ASCII Decimal to 24-Bit Binary	: D$UFRNUM

ufr	***$CVB24L***	Convert ASCII Decimal to 24-Bit Binary	: D$UFRNUM
ufr	***$CVD***	Convert 16-Bit Binary to ASCII Decimal	: D$UFRNUM
ufr	***$CVD24***	Convert 24-Bit Binary to Decimal	: D$UFRNUM
ufr	***$CVD24Z***	$CVD24 with Zero-Suppression	: D$UFRNUM
ufr	***$CVDZ***	$CVD with Zero-Suppression	: D$UFRNUM
typ	***$CVSTBL***	STRUCT;$CVSTIME Output Area	: D$UFRGEN
fld	***$CVSTDAY***	BYTE; Day of Month	: $CVSTBL
fld	***$CVSTDOW***	BYTE; Day of Week	: $CVSTBL
fld	***$CVSTHH***	BYTE; Hour of Day	: $CVSTBL
ufr	***$CVSTIME***	Obtain System Date,Time Info	: D$UFRGEN
fld	***$CVSTJD***	UNSIGNED; Julian Date (Day of Year)	: $CVSTBL
fld	***$CVSTMM***	BYTE; Minute of Hour	: $CVSTBL
fld	***$CVSTMON***	BYTE; Month of Year	: $CVSTBL
fld	***$CVSTSS***	BYTE; Second of Minute	: $CVSTBL
fld	***$CVSTYR***	UNSIGNED; Year	: $CVSTBL

var	***D$A***	EXTERN D$A BYTE;	: D$INC
var	***D$B***	EXTERN D$B BYTE;	: D$INC
var	***D$BC***	EXTERN D$BC UNSIGNED;	: D$INC
var	***D$BUF1***	EXTERN D$BUF1 [256] BYTE;	: D$INC
var	***D$BUF2***	EXTERN D$BUF2 [256] BYTE;	: D$INC
var	***D$BUF3***	EXTERN D$BUF3 [256] BYTE;	: D$INC
var	***D$BUF4***	EXTERN D$BUF4 [256] BYTE;	: D$INC
var	***D$BUF5***	EXTERN D$BUF5 [256] BYTE;	: D$INC
var	***D$C***	EXTERN D$C BYTE;	: D$INC
def	***D$CALL***	Call an Arbitrary Extern Defined Subroutine	: D$INC
typ	***D$CALLF***	(); External Function to CALL Subroutines	: D$INC
var	***D$CC***	EXTERN D$CC D$CCODE;	: D$INC
typ	***D$CCODE***	Condition Code Flags	: D$INC
val	***D$CFLAG***	001 Carry flag : D$CCODE	: D$INC
def	***D$COMP***	Block Compare Two Character Strings	: D$INC
var	***D$D***	EXTERN D$D BYTE;	: D$INC
var	***D$DE***	EXTERN D$DE UNSIGNED;	: D$INC
var	***D$E***	EXTERN D$E BYTE;	: D$INC
inc	***D$ERRCODE***	System Error Code	
inc	***D$ERRNUM***	RMS System Function Error Class Numbers	
inc	***D$FAR***	File Access Routines, REQUIRES D$RMS	
def	***D$GET24***	Numeric, Convert 24 Bit to 32 Bit Value	: D$INC
var	***D$H***	EXTERN D$H BYTE;	: D$INC
var	***D$HL***	EXTERN D$HL UNSIGNED;	: D$INC
inc	***D$INC***	Basic DASL definitions, Standard Include	
def	***D$INFO***	Return Processor Type	: D$INC
def	***D$JUMP***	EXTERN (f ∧ D$CALLF)	: D$RMS
var	***D$L***	EXTERN D$L BYTE;	: D$INC
def	***D$MOVE***	Block Move 0 to 65535 Bytes	: D$INC
def	***D$MOVER***	Block Move Reverse, Starting at Ending Adr	: D$INC
inc	***D$PCR***	Program Communications Region Definitions	
val	***D$PFLAG***	010 Parity flag : D$CCODE	: D$INC
def	***D$PUT24***	Numeric, Convert 32 Bit to 24 Bit Value	: D$INC
def	***D$RFI2***	DASL external function for defining macros in	
	D$RFI2	include files (not a user function).	: D$RMSPROG
inc	***D$RMS***	Common Nucleus and UFR definitions	
inc	***D$RMSGEN***	RMS General System Function Definitions	
inc	***D$RMSIO***	File Handling, Block I/O, Disk, Printer, Pipe	

inc	D$RMSMEM	Memory Management Definitions	
inc	D$RMSPROG	Program Loading and Execution Control	
inc	D$RMSSPEC	Special System Calls	
inc	D$RMSSTRUCT	Disk Structure Definitions	
inc	D$RMSTASK	User Multi-tasking	
inc	D$RMSWS	Workstation	
def	D$SC	Call a SYSTEM CALL External Function	: D$INC
val	D$SFLAG	004 Sign flag	: D$CCODE : D$INC
inc	D$UFRENV	Environment Handling	
inc	D$UFRENV	Requires D$RMSIO include if $$FILENAM function used	
inc	D$UFRERR	Error Handling	
inc	D$UFRGEN	General Utility	
inc	D$UFRLIB	Library Manipulation	
inc	D$UFRMEM	Memory Management	
inc	D$UFRNQDQ	NQ/DQ UFR Definitions	
inc	D$UFRNUM	Numeric Manipulation	
inc	D$UFRRLD	Relocating Loader	
inc	D$UFRSCAN	Command Interpreter (Scanning)	
inc	D$UFRSYS	System Interface	
inc	D$UFRWFIO	Work File I/O	
inc	D$UFRWS	Workstation Interface	
inc	D$WORKSTN	Special Workstation Definitions	|*
var	D$X	EXTERN D$X BYTE;	: D$INC
var	D$XA	EXTERN D$XA UNSIGNED;	: D$INC
val	D$ZFLAG	002 Zero flag	: D$CCODE : D$INC
abr	DASL	Datapoint Advanced Systems Language	
fld	DATA	[0] CHAR; Environment data	: $ENVT
ufr	$DATETIM	Convert System Time to Standard	: D$UFRGEN
var	DBLBUFF$	EXTERN DBLBUFF$ BYTE; Driver Routine ?:	D$UFRWFIO
abr	DBMS	Data Base Management System	
abr	DBOMP	DATABUS Compiler	
typ	$DCB	STRUCT; Data File Control Block	: D$FAR
fld	$DCBACC	BYTE; $ACCODES file access flags ($AC..)	: $DCB
fld	$DCBAPFD	∧ $PFDB; Alternate PFDB pointer	: $DCB
fld	$DCBBLK	BYTE; Block size from $MFDBLSZ	: $DCB
fld	$DCBCBFP	BYTE; Current buffer	: $DCB
fld	$DCBCLFP	$FILEPTR; Current (real) file ptr (LSB..MSB)	: $DCB
fld	$DCBCLRP	$FILEPTR; Logical record ptr (LSB..MSB)	: $DCB
fld	$DCBCPFD	∧ $PFDB; Current PFDB pointer	: $DCB
fld	$DCBEOFP	$FILEPTR; EOF pointer	: $DCB
fld	$DCBFLG1	BYTE; DFCB flag byte 1	: $DCB
fld	$DCBFLG2	BYTE; DFCB flag byte 2	: $DCB
fld	$DCBHBFP	BYTE; High dirty buffer pointer	: $DCB
fld	$DCBMAB	$MAB; Managed DCB redefinition of PFDB	: $DCB
fld	$DCBPFDB	$PFDB; Data file PFDB	: $DCB
far	$DCLOSE	Direct Access, Close	: D$FAR
abr	DCR	?	
far	$DDEL	Direct Random Access, Delete	: D$FAR
far	$DDELCR	Direct Sequential Access, Delete Current	: D$FAR
ufr	$DECOHSI	Environment, Decompress an HSI String	: D$UFRENV
dcw	DEFAULT	Preceeds Statement in CASE for No-match Condition	
dcm	DEFINE	Define a String to be Substituted for Identifier	
fld	$DELIMT	[16] BYTE; Name delimiter characters	:$INFOITEM
val	$DELMENV	072 ':' Environment name follows	:D$UFRSCAN
val	$DELMEXT	057 '/' File extension follows	:D$UFRSCAN

val	$DELMHSI	056 '.' HSI level delimiter	:D$UFRSCAN
val	$DELMNOT	055 '-' Not mark	:D$UFRSCAN
val	$DELMOPQ	042 '"' Option quoted value delimiter	:D$UFRSCAN
val	$DELMOPT	073 ';' Command line options delimiter	:D$UFRSCAN
val	$DELMORE	053 '+' More mark	:D$UFRSCAN
val	$DELMQRY	077 '?' Query mark	:D$UFRSCAN
val	$DELMST1	054 ',' Valid specification terminator	:D$UFRSCAN
val	$DELMSYM	075 '=' Symbolic field name preceeds and	:D$UFRSCAN
abr	DFCB	Data File Control Block	
far	$DGETCRK	Direct Seq Access, Get Current Record Key	: D$FAR
far	$DGETNRK	Direct Seq Access, Get Next Record Key	: D$FAR
abr	DID	Destination Identification	
far	$DIOCLR	Direct Access, I/O Clear	: D$FAR
sc	$DISCONT	Disconnect Node	: D$RMSIO
fld	$DISKEY	UNSIGNED; Displacement of key within entry	: $DISTBL
fld	$DISKEYL	UNSIGNED; Length of key	: $DISTBL
fld	$DISNUM	UNSIGNED; Number of table entries	: $DISTBL
ufr	$DISORT	Diminishing Increment In-Core Sort	: D$UFRGEN
ufr	$DISPCH	Workstn-IF, Display One Character	: D$UFRWS
fld	$DISRLEN	UNSIGNED; Length of table	: $DISTBL
fld	$DISTAB	∧ BYTE; Location of table	: $DISTBL
typ	$DISTBL	STRUCT; $DISORT Parameter Table	: D$UFRGEN
fld	$DISWORK	∧ BYTE; Location of work area	: $DISTBL
ufr	$DIVID3	Numeric, Unsigned 24-bit Division	: D$UFRNUM
val	$DK863M	0015 8600 3M Cartridge tape :$OPENPT	: D$RMS
val	$DK883M	0014 8800 3M Cartridge tape :$OPENPT	: D$RMS
val	$DKCARDP	0011 Card punch :$OPENPT	: D$RMS
val	$DKCARDR	0010 Card reader :$OPENPT	: D$RMS
val	$DKCASS	0004 Cassette tape :$OPENPT	: D$RMS
val	$DKCOMM	0006 Communications channel :$OPENPT	: D$RMS
val	$DKDISK	0001 Disk :$OPENPT	: D$RMS
val	$DKFAX	020 FAX Equipment :$OPENPT	: D$RMS
val	$DKMAGT	5 Industry compatible mag tape :$OPENPT	: D$RMS
val	$DKMAX	DEFINEd 020 Largest resource kind number	: D$RMSIO
val	$DKPIPE	0002 Pipe (soft resource) :$OPENPT	: D$RMS
val	$DKPRINT	0003 Printer :$OPENPT	: D$RMS
val	$DKPTP	0013 Paper tape punch :$OPENPT	: D$RMS
val	$DKPTR	0012 Paper tape reader :$OPENPT	: D$RMS
val	$DKRIM	0017 Direct RIM access :$OPENPT	: D$RMS
val	$DKSMPLR	016 Task executn time sampler :$OPENPT	: D$RMS
val	$DKTIMER	0007 Delay timer clk,soft res :$OPENPT	: D$RMS
val	$DKWS	0000 Work station (pseudo res) :$OPENPT	: D$RMS
abr	DLL	Down Line Load	
ufr	$DLMCHEK	Check Character For a Delimiter	: D$UFRWS
abr	DLMT	Delimiter	
sc	$DONATFV	Donate a FAV to a specified task	: D$RMSSPEC |*
far	$DOPEN	Direct Access, Open	: D$FAR
abr	DOS	Disk Operating System	
far	$DPOS	Direct Random Access, Position	: D$FAR
far	$DPOSEOF	Direct Random Access, Position to EOF	: D$FAR
far	$DPOSNX	Direct Sequential Access, Position to Next	: D$FAR
far	$DPOSPV	Direct Sequential Access, Pos to Previous	: D$FAR
far	$DREAD	Direct Random Access, Read	: D$FAR
far	$DREADCR	Direct Sequential Access, Read Current	: D$FAR
far	$DREADNX	Direct Sequential Access, Read Next	: D$FAR

far	***$DREADPV***	Direct Sequential Access, Read Previous	: D$FAR
far	***$DRWRT***	Direct Random Access, Rewrite	: D$FAR
far	***$DRWRTCR***	Direct Sequential Access, Rewrite Current	: D$FAR
abr	***DST***	Daylight Savings Time	
typ	***$DSTINFO***	Daylight Savings time Start/Stop Table	: D$RMSGEN
far	***$DWRITE***	Direct Random Access, Write	: D$FAR
far	***$DWRITNX***	Direct Sequential Access, Write Next	: D$FAR
far	***$DWRTEOF***	Direct Sequential Access, Write End of File	: D$FAR

NOTE: In the *FUNCTIONS* section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually *SC$...* <u>or</u> *$UEC...* <u>and</u>
$ERRC.$CODE is usually *$EC...nn* <u>or</u> *$UEC...nn*

The ***$EC...nn*** or ***$UEC...nn*** ends with a decimal number which is its value.

The words with a ***:**** at the end of the line in this section, ***WORDS***, are defined in ***DASL*** in the ***D$ERRCODE*** include file. You may use those words in statements testing the value of ***$ERRC.$CODE***, otherwise you must use the decimal value.

val ***$ECDFV2***	2	Specified task does not exist	:*
val ***$ECFMS0***	0	FCB not open	:*
val ***$ECFMS1***	1	Invalid open mode requested	:*
val ***$ECFMS2***	2	Invalid FCB type for operation	:*
val ***$ECFMS3***	3	Open attempted on an open FCB	:*
val ***$ECFMS4***	4	No remaining address space	:*
val ***$ECFMS5***	5	Write-protected sector mapped	:*
val ***$ECFMS6***	6	$FCBNBFS is zero	:*
val ***$ECFMS7***	7	Requested resource not disk	:*
val ***$ECFMS8***	8	*** RESERVED FOR FUTURE USE ***	:*
val ***$ECFMS9***	9	Invalid character in output record	:*
val ***$ECFMS10***	10	Record format error	:*
val ***$ECFMS11***	11	Invalid index file type	:*
val ***$ECFMS12***	12	Insufficient buffers for ISAM block	:*
val ***$ECFMS13***	13	Environment entry for ISAM data file not found	:*
val ***$ECFMS14***	14	Illegal operation -- duplicate record	:*
val ***$ECFMS15***	15	Invalid LFV type in FMT	:*
val ***$ECFMS16***	16	Pipe message sequence error from FMT	:*
val ***$ECFMS17***	17	Log-on device not a pipe	:*
val ***$ECFMS18***	18	Index file env in MFD not found	:*
val ***$ECFMS19***	19	Data file env in MFD not found	:*
val ***$ECFMS20***	20	Invalid open type for direct file	:*
val ***$ECFMS21***	21	No FMT pipes found	:*
val ***$ECFMS22***	22	Buffer attempt beyond $BPLAST	:*
val ***$ECFMS23***	23	Bytes expected greater than bytes in response	:*
val ***$ECFMS24***	24	Response byte count exceeded	:*
val ***$ECFMS25***	25	$DOPEN attempted on indexed MFD	:*
val ***$ECFMS26***	26	Index key length exceeds FCB key length	:*
val ***$ECFMS27***	27	Data file is MFD type	:*
val ***$ECFMS28***	28	MFD file version incompatible	:*
val ***$ECFMS29***	29	FMT version incompatible	:*
val ***$ECFMS30***	30	Managed file access violation	:*

NOTE: In the ***FUNCTIONS*** section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually ***SC$...*** or ***$UEC...*** and
$ERRC.$CODE is usually ***$EC...nn*** or ***$UEC...nn***

The ***$EC...nn*** or ***$UEC...nn*** ends with a decimal number which is its value.

The words with a ***:**** at the end of the line in this section, ***WORDS***, are defined in ***DASL*** in the ***D$ERRCODE*** include file. You may use those words in statements testing the value of ***$ERRC.$CODE***, otherwise you must use the decimal value.

val	***$ECFMS31***	31	FMT connection lost	:*
val	***$ECFMS32***	32	Unmanaged FAR not in user command file	:*
val	***$ECFMS33***	33	Managed FAR not in user command file	:*
val	***$ECFMS35***	35	Data read access violation	:*
val	***$ECFMS36***	36	Data write access violation	:*
val	***$ECFMS37***	37	Index read access violation	:*
val	***$ECFMS38***	38	Index write access violation	:*
val	***$ECFMS39***	39	$IPREP: invalid $FCBBLKL	:*
val	***$ECFMS40***	40	$IPREP: invalid $FCBKLGT	:*
val	***$ECFMS46***	46	data file not in text format	:*
val	***$ECFMS46***	46	Data file is not in text format	:*
val	***$ECFMS47***	47	Environment entry for data file not found	:*
val	***$ECFMS48***	48	Insufficient buffers for AIM index	:*
val	***$ECFMS49***	49	Bad AIM index	:*
val	***$ECFMS50***	50	Insufficient data in key	:*
val	***$ECFMS51***	51	Key conflict	:*
val	***$ECFMS52***	52	Incorrect key format	:*
val	***$ECFMS53***	53	No valid read prior to $areadkg	:*
val	***$ECFMS54***	54	Illegal $ARWRTCR or $ADELCR	:*
val	***$ECFMS55***	55	Invalid free-float key specification	:*
val	***$ECFMS57***	57	Invalid data file cursor	:*
val	***$ECFMS58***	58	Invalid maxi associated data file spec	:*
val	***$ECFMS59***	59	Incompatible AIM index	:*
val	***$ECFMT0***	0	No more UAVs	:*
val	***$ECFMT1***	1	No more LFVs	:*
val	***$ECFMT2***	2	No more AFVs	:*
val	***$ECFMT3***	3	Invalid version number in config rec	:*
val	***$ECFMT4***	4	Invalid processor for configuration	:*
val	***$ECFMT5***	5	No such debug ws	:*
val	***$ECFMT6***	6	Debug ws name missing	:*
val	***$ECFMT7***	7	No more logical address space	:*
val	***$ECFMT8***	8	FMT root module invalid	:*
val	***$ECFMT9***	9	Overlay descriptor not initialized	:*
val	***$ECFMT10***	10	Transfer address not given ($OVLRET)	:*
val	***$ECFMT11***	11	Invalid $BPFLAG field in message	:*
val	***$ECFMT12***	12	Invalid request message length	:*
val	***$ECFMT13***	13	Sequence error	:*
val	***$ECFMT14***	14	Invalid operation code in req	:*
val	***$ECFMT15***	15	Invalid working set (MOVDPT$)	:*
val	***$ECFMT16***	16	Req msg format error	:*

val	*$ECFMT17*	17	File access violation	:*
val	*$ECFMT18*	18	Invalid close mode	:*
val	*$ECFMT19*	19	Invalid LFV type for operation	:*
val	*$ECFMT20*	20	Invalid LFV id	:*
val	*$ECFMT21*	21	Invalid record size requested	:*
val	*$ECFMT22*	22	Invalid block size requested at open	:*
val	*$ECFMT23*	23	Invalid open mode requested	:*
val	*$ECFMT24*	24	Attempted open on non-disk resource	:*
val	*$ECFMT25*	25	Attempted open on MFD file	:*
val	*$ECFMT26*	26	Attempted read past EOF (PREAD$)	:*
val	*$ECFMT27*	27	ISAM data file name does not match	:*
val	*$ECFMT28*	28	ISAM data file mask does not match	:*
val	*$ECFMT29*	29	Invalid ISAM index file format	:*
val	*$ECFMT30*	30	SQL too large for prep or create	:*
val	*$ECFMT31*	31	No user pipe connection	:*
val	*$ECFMT33*	33	FMT locked	:*
val	*$ECFMT34*	34	Attempted config on file with log ext	:*
val	*$ECFMT35*	35	Invalid value for timing option (FMTCONT)	:*
val	*$ECFMT39*	39	Invalid default configuration (CONFGFMT)	:*
val	*$ECFMT40*	40	Invalid request code	:*
val	*$ECFMT41*	41	Invalid configuration file (CONFGFMT)	:*
val	*$ECLIO0*	0	Illegal operation	
val	*$ECLIO1*	1	LFDB not open	
val	*$ECLIO2*	2	Attempt to write an illegal byte	
val	*$ECLIO4*	4	Record out of range	
val	*$ECLIO6*	6	Attempt to rewrite a long record	
val	*$ECLIO8*	8	Logical rewrite attempted on a deleted record	
val	*$ECLIO9*	9	Terminating adr less than file-cursor in $LRPDE	
val	*$ECLIO10*	10	ISAM open attempted on non-disk resource	
val	*$ECLIO11*	11	Invalid close mode requested	
val	*$ECLIO12*	12	PDAM module for resource not configured	
val	*$ECLIO13*	13	Invalid open mode requested	
val	*$ECLIO14*	14	User abort in exception routine	
val	*$ECLIO20*	20	Too many levels of index - reorganize index	
val	*$ECLIO21*	21	PDAM $PMXBF less than block size	
val	*$ECLIO22*	22	ISAM index structure fault	
val	*$ECLIO23*	23	Duplicate keys not allowed	
val	*$ECLIO24*	24	Key length greater than $IKEYLEN	
val	*$ECLIO25*	25	Key length exceeds limit for block size	
val	*$ECLIO26*	26	Key length exceeds ISAM limit	
val	*$ECLIO27*	27	Index key length exceeds IFDB key length	
val	*$ECLIO30*	30	Operation timed out	
val	*$ECLIO31*	31	Communications link failure	
val	*$ECLIO32*	32	Break received	
val	*$ECLIO33*	33	Data received without available buffer	

NOTE: In the *FUNCTIONS* section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually *SC$...* or *$UEC...* and
$ERRC.$CODE is usually *$EC...nn* or *$UEC...nn*

The *$EC...nn* or *$UEC...nn* ends with a decimal number which is its value.

The words with a *:** at the end of the line in this section, *WORDS*, are defined in *DASL* in the *D$ERRCODE* include file. You may use those words in statements testing the value of *$ERRC.$CODE*, otherwise you must use the decimal value.

val	*$ECLIO34*	34	Parity error on received data	
val	*$ECLIO35*	35	Lost carrier	
val	*$ECLIO36*	36	Requested function not available	
val	*$ECLIO37*	37	Existing connection still present	
val	*$ECLIO38*	38	No power-on indication from A.C.U.	
val	*$ECLIO39*	39	A.C.U. malfunctioned	
val	*$ECLIO40*	40	A.C.U. retry limit exceeded	
val	*$ECLIO41*	41	Invalid character in phone number string	
val	*$ECLIO50*	50	Invalid sub-function requested	
val	*$ECLIO51*	51	Record format error	
val	*$ECLIO52*	52	$FDTFLEN points to an LSN containing no EOF mark	
val	*$ECLIO70*	70	File on tape not found	
val	*$ECLIO71*	71	Invalid tape section mounted	
val	*$ECLIO72*	72	Invalid standard level in ANSI VOL1 label	
val	*$ECLIO73*	73	Invalid label type encountered	
val	*$ECLIO74*	74	File on tape not expired	
val	*$ECLIO75*	75	Record format error on tape	
val	*$ECLIO76*	76	Invalid block count found in trailer label	
val	*$ECLIO77*	77	Invalid $TPUTEOV request for opened label-set	
val	*$ECLIO78*	78	Not enough buffers to contain maximum block	
val	*$ECLIO79*	79	Invalid tape resource	
val	*$ECLIO80*	80	Missing VOL1 label	
val	*$ECLIO81*	81	Invalid max. block size found in the HDR2 label	
val	*$ECLIO82*	82	Invalid record size found in the HDR2 label	
val	*$ECLIO83*	83	Missing EOV/EOF label-set	
val	*$ECLIO84*	84	Invalid label-set requested at open	
val	*$ECLIO85*	85	Label-set requested at open is not configured	
val	*$ECLIO86*	86	Missing HDR2 label	
val	*$ECLIO87*	87	Too many user labels encountered	
val	*$ECLKF2*	2	Invalid mode	:*
val	*$ECLOAD2*	2	Read error in absolute code file	:*
val	*$ECLOAD3*	3	Format error in absolute code file	:*
val	*$ECLOAD4*	4	Insufficient user logical address space	:*
val	*$ECLOAD6*	6	Invalid absolute code file format	:*
val	*$ECLOAD7*	7	Read error in absolute code file	:*
val	*$ECLOAD8*	8	Load address outside user mapped space	:*
val	*$ECLPS0*	0	Bad parameter	
val	*$ECLPS1*	1	Bad Zone Number	
val	*$ECLPS2*	2	Release LMT	
val	*$ECLPS3*	3	No Physical Memory	

val	***$ECLPS4***	4	Bad Parameters for LCALL	
val	***$ECLPS5***	5	LPS Software Stack Under/Overflow	
val	***$ECLPS6***	6	LPS Out of Memory	
val	***$ECLPS7***	7	Patching -- External Memory Request	
val	***$ECLPS8***	8	Patching -- LMT Error	
val	***$ECLPS9***	9	Bad Processor (not 8600 or 8800)	
val	***$ECMCTL1***	1	Invalid function number	:*
val	***$ECMCTL2***	2	Not on console with maximum security level	:*
val	***$ECMCTL3***	3	No room in System Table	:*
val	***$ECMGET0***	0	No more memory sectors available	:*
val	***$ECMGET1***	1	Already at maximum allocation	:*
val	***$ECMKEY0***	0	Invalid mapped sector number entry	:*
val	***$ECMKEY1***	1	Memory sector not allocated	:*
val	***$ECMMAP0***	0	Invalid physical sector key entry	:*
val	***$ECMMAP1***	1	Invalid mapped sector number entry	:*
val	***$ECMREL0***	0	Invalid physical sector key entry	:*
val	***$ECMREL1***	1	Shared read-only mem sector release attempted	:*
val	***$ECMREL2***	2	PCR memory sector release attempted	:*
val	***$ECMREL3***	3	Specified memory sector has I/O in progress	:*
val	***$ECMTX0***	0	User connection lost	:*
val	***$ECMTX1***	1	Request message length error	:*
val	***$ECMTX2***	2	FMT is locked due to "FMTCONT..;die"	:*
val	***$ECMTX3***	3	Message buffer sequence error	:*
val	***$ECMTX4***	4	Invalid operation	:*
val	***$ECMTX5***	5	No more users can be handled by this FMT	:*
val	***$ECMTX6***	6	Workstation for "FTMCONT..;DEBUG=" not avail	:*
val	***$ECMTX7***	7	Invalid configuration file version	:*
val	***$ECMTX8***	8	Invalid processor type	:*
val	***$ECMTX9***	9	Out of address space (global initialization)	:*
val	***$ECMTX10***	10	Out of addr space (MTX member initializatn)	:*
val	***$ECMTX11***	11	MTXDOWN$ internal error	:*
val	***$ECMTX12***	12	No such MTX member	:*
val	***$ECMTX13***	13	GETBPSK$ (buffer management) internal error	:*
val	***$ECMTX50***	50	Address 010000 already mapped ($MTXUP)	:*
val	***$ECMTX51***	51	Configuration file already open	:*
val	***$ECMTX52***	52	No current DCR selected	:*
val	***$ECMTX53***	53	Configuration file must not have extent "LOG"	:*
val	***$ECRFI0***	0	Invalid interrupt type number	:*
val	***$ECRFI1***	1	Workstation required	:*
val	***$ECSCL1***	1	Invalid function mode	:*
val	***$ECSCL2***	2	Invalid test output type	:*
val	***$ECSCL3***	3	Selective test output list full	:*

NOTE: In the *FUNCTIONS* section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually *SC$...* <u>or</u> *$UEC...* <u>and</u>
$ERRC.$CODE is usually *$EC...nn* <u>or</u> *$UEC...nn*

The *$EC...nn* or *$UEC...nn* ends with a decimal number which is its value.

The words with a *:** at the end of the line in this section, ***WORDS***, are defined in ***DASL*** in the ***D$ERRCODE*** include file. You may use those words in statements testing the value of *$ERRC.$CODE*, otherwise you must use the decimal value.

val	*$ECSCL4*	4	Function not available	:*
val	*$ECSCL5*	5	No console privileges	:*
val	*$ECSCL6*	6	Specified task does not exist	:*
val	*$ECSCL7*	7	Not on physical system console	:*
val	*$ECSCL8*	8	Invalid line number	:*
val	*$ECSIO01*	01	Invalid file access variable identificatn key	:*
val	*$ECSIO02*	02	Invalid physical sector key in parameter table	:*
val	*$ECSIO03*	03	Invalid buffer address in parameter table	:*
val	*$ECSIO04*	04	Invalid "to do" value in parameter table	:*
val	*$ECSIO05*	05	Resource no longer available	:*
val	*$ECSIO06*	06	Hard resource error(media/contrllr)-get help!	:*
val	*$ECSIO07*	07	File access violation	:*
val	*$ECSIO08*	08	Read beyond end of physical file attempted	:*
val	*$ECSIO09*	09	Resource write protected	:*
val	*$ECSIO10*	10	Operation still in progress on file access var	:*
val	*$ECSIO11*	11	All available segments in file have been used	:*
val	*$ECSIO12*	12	Hard disk err while using sys tables-get help!	:*
val	*$ECSIO13*	13	Hard disk err during disk structr change-help!	:*
val	*$ECSIO14*	14	Disk structure (system table) error-get help!	:*
val	*$ECSIO15*	15	Disk write protected (structure unchanged)	:*
val	*$ECSIO16*	16	Operation timed out	:*
val	*$ECSIO17*	17	Operation aborted by user	:*
val	*$ECSIO18*	18	Invalid sub-func code given in param. table	:*
val	*$ECSIO19*	19	More data read than can fit in specified buffr	:*
val	*$ECSIO20*	20	No I/O was outstanding	:*
val	*$ECSIO21*	21	Node given in specified env does not respond	:*
val	*$ECSIO22*	22	Resource given in specified env is not online	:*
val	*$ECSIO23*	23	File could not be found using specified env	:*
val	*$ECSIO24*	24	Required catalog file does not exist	:*
val	*$ECSIO25*	25	File or resource busy with other use	:*
val	*$ECSIO26*	26	No more space on disk	:*
val	*$ECSIO27*	27	No more space in disk directory	:*
val	*$ECSIO28*	28	System table space exhausted	:*
val	*$ECSIO29*	29	No overlay space for resource driver	:*
val	*$ECSIO30*	30	Name contains invalid characters	:*
val	*$ECSIO31*	31	Name already in use	:*
val	*$ECSIO32*	32	Logical sector number outside allocated space	:*
val	*$ECSIO33*	33	Rename to different resource attempted	:*
val	*$ECSIO34*	34	Invalid mode	:*
val	*$ECSIO35*	35	Item not found	:*

```
val $ECSIO36    36 Too many items found                        :*
val $ECSIO37    37 Invalid access bit supplied                 :*
val $ECSIO38    38 Invalid file descriptn table phys cluster nr  :*
val $ECSIO40    40 Too many passwords supplied                 :*
val $ECSIO41    41 Invalid time value                          :*
val $ECSIO42    42 Security change access can't be removed from
                   file                                        :*
val $ECSIO43    43 Named driver overlay not included in config   :*
val $ECSIO44    44 Data communication system failure           :*
val $ECSIO45    45 Remote connection failure                   :*
val $ECSIO46    46 Operation beyond end of resource attempted  :*
val $ECSIO47    47 Feed command not accepted                   :*
val $ECSIO48    48 Multi-punch error                           :*
val $ECSIO49    49 Attmpt made to create a pipe w/o delete access:*
val $ECSIO50    50 A file is open on that node                 :*
val $ECSIO51    51 Mem access violation at  other end of a pipe  :*
val $ECSIO52    52 Remote pipe req hooked to another remote req  :*
val $ECSIO53    53 Multifile resource reserved                 :*
val $ECSIO54    54 Remote operation not supported              :*
val $ECSIO55    55 Node given not configurd for file node support:*
val $ECSIO56    56 Net name needd if node linkd to 2 or more nets
val $ECSIO57    57 Driver cannot execute with current config   :*
val $ECSIO58    58 Invalid hash code in bad track table        :*
val $ECSIO59    59 Driver not successfully loaded into IMOD    :*
val $ECSIO60    60 Specified file is not a catalog file        :*
val $ECSIO61    61                            ?                :*
val $ECSIO62    62 Power lost to printer                       :*
val $ECSIO63    63 Resource name too long                      :*
val $ECSIO64    64 HSI data too long                           :*
val $ECSMAX0     0 Maximum must not be less than current minimum :*
val $ECSMAX1     1 Max must not be less than current allocation  :*
val $ECSMIN0     0 Function invalid in shared program          :*
val $ECSMIN1     1 Insufficient memory to grant request        :*
val $ECSTM1      1 No console privileges                       :*
val $ECSTM2      2 Invalid time table                          :*  |*
val $ECSTM3      3 Invalid mode                                :*  |*
val $ECTCL0      0 Invalid task identifier supplied            :*
val $ECTCL1      1 No access to specified task                 :*
val $ECTCL2      2 Invalid task secret number supplied         :*
val $ECTCL3      3 Invalid mode                                :*
val $ECTERM      DEFINE'd MAXUNSIGNED                 : D$ERRNUM
    $ECTERM      0177777 nevr possible err code,msg file terminator
val $ECTRAP0     0 Invalid interrupt type number               :*
val $ECTSK2      2 Memory donated as PCR has outstanding I/O   :*
val $ECTSK3      3 Task creation attempted by a local task     :*
val $ECTSK4      4 No sys table space for new user task ctrl blck:*
val $ECTSK5      5 Specified task does not exist
val $ECTSK5      5 Invalid task identifier supplied
```

NOTE: In the *FUNCTIONS* section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually *SC$...* <u>or</u> *$UEC...* <u>and</u>
$ERRC.$CODE is usually ***$EC...nn*** <u>or</u> ***$UEC...nn***

The ***$EC...nn*** or ***$UEC...nn*** ends with a decimal number which is its value.

The words with a ***:**** at the end of the line in this section, ***WORDS***, are defined in ***DASL*** in the ***D$ERRCODE*** include file. You may use those words in statements testing the value of *$ERRC.$CODE*, otherwise you must use the decimal value.

val	***$ECTSK6***	6	Indepndnt task termination by task not allowed	:*	
val	***$ECTSK6***	6	Access violation	:*	
val	***$ECTSK7***	7	Invalid task secret number supplied	:*	
val	***$ECTSK8***	8	Insufficient memory or program address space	:*	
val	***$ECTSK9***	9	Invalid mode	:*	
val	***$ECTSK11***	11	Format error during indep. task program load	:*	
val	***$ECTSK12***	12	Read error during indep. task program load	:*	
val	***$ECTSK13***	13	Not enough physical memory for indep task prgm	:*	
val	***$ECTSK14***	14	Not enough log addr space for indep task prgm	:*	
val	***$ECTSK15***	15	Memory access viol. during indp task prgm load	:*	
val	***$ECTSK16***	16	Dual sector table unsupported for ind.tsk prgm	:*	
val	***$ECTSK17***	17	Shared prgm error during indep task prgm load	:*	
val	***$ECTSK18***	18	No room in system table for new SPV	:*	
val	***$ECTSK29***	29	No room for new user task	:*	|*
val	***$ECUAB0***	0	User ABEND already active		
val	***$ECUAB2***	2	User ABEND not active	:*	
val	***$ECUAB3***	3	Specified task does not exist	:*	
val	***$ECUAB4***	4	No access to task to set User ABEND	:*	
val	***$ECUAB5***	5	No access to task to set USER ABEND	:*	
val	***$ECUAB6***	6	Resource is a byte string device	:*	
val	***$ECUCS0***	0	Invalid usr creatd semaphore identifier suppld	:*	
val	***$ECUCS1***	1	UCS not owned by this task, wrong father	:*	
val	***$ECUCS2***	2	Closed semaphores cannot be deleted	:*	
val	***$ECUCSG0***	0	No systm tabl space for new usr creatd semphor	:*	
val	***$ECUMAV***	0	memory access violation	: D$ERRNUM	
val	***$ECWIO01***	1	No expanded function keyboard available	:*	|*
val	***$ECWSCC1***	1	Workstation off line	:*	
val	***$ECWSCC2***	2	Invalid cursor position	:*	
val	***$ECWSCC3***	3	Invalid mode	:*	
val	***$ECWSGC0***	0	Keyboard entry fifo empty	:*	
val	***$ECWSGC1***	1	Workstation off line	:*	
val	***$ECWSIO1***	1	Workstation off line	:*	
val	***$ECWSIO2***	2	Invalid cursor position	:*	
val	***$ECWSIO3***	3	Invalid control function code	:*	
val	***$ECWSIO4***	4	Keyin aborted by function key	:*	
val	***$ECWSIO5***	5	Keyin aborted due to time out	:*	
val	***$ECWSIO6***	6	Invalid control string parameter	:*	
val	***$ECWSTA1***	1	Workstation off-line	:*	
val	***$ECWSWT1***	1	Workstation off-line	:*	

val	*$EEOF*	0200 Erase from cursor to end of frame	: D$RMS
val	*$EEOL*	0201 Erase from cursor to end of line	: D$RMS
val	*$EL*	0232 Advance to new line,terminate strg	: D$RMSWS
abr	*ELF*	Error Logging File	
dcw	*ELSE*	Part of IF THEN ELSE Execution Control	
abr	*ENQ*	Enquiry	
ddw	*ENTRY*	Declare Global Name; may be Referenced Externally	
dim	*ENUM*	Define Var Type BYTE, Values 0 thru 8	: D$INC
dim	*ENUMV*	Define Values Incrementing from Initial	: D$INC
abr	*ENV*	Environment	
fld	*$ENV*	$ENVN; Environment	: $NAMEEXTENV
ufr	*$ENVDEL*	Delete Existing Environment	: D$UFRENV
ufr	*$ENVFNDM*	Find Environment Data Match	: D$UFRENV
ufr	*$ENVINS*	Insert New Environment	: D$UFRENV
ufr	*$ENVLGET*	Obtain Environment Entry Length	: D$UFRENV
ufr	*$ENVLOC*	Locate Existing Environment	: D$UFRENV
val	*$ENVMAXL*	(SIZEOF<$ENVT>)+3*((SIZEOF<$NAMET>)+1)+(SIZEOF<$HSI>)	
	$ENVMAXL	000313	: D$UFRENV
typ	*$ENVN*	[8] CHAR; Environment Name	: D$RMS
ufr	*$ENVPDAT*	Position to Env Data in an Env Entry	: D$UFRENV
ufr	*$ENVPEEL*	Create Master Catalog Environment	: D$UFRENV
ufr	*$ENVPHSI*	Pos to HSI Name in an Environment Entry	: D$UFRENV
ufr	*$ENVPLOP*	Pos to UET Link In Open Parameter Table	: D$UFRENV
ufr	*$ENVPNAM*	Pos to Environment Name in an Env Entry	: D$UFRENV
ufr	*$ENVPNET*	Position to Net Name in an Env Entry	: D$UFRENV
ufr	*$ENVPNOD*	Position to Node Name in an Env Entry	: D$UFRENV
ufr	*$ENVPPAS*	Position to First Password in Env Entry	: D$UFRENV
ufr	*$ENVPRES*	Position to Resource Name in Env Entry	: D$UFRENV
typ	*$ENVT*	STRUCT; Environment Table Entry	: D$UFRENV
val	*$ENVTERM*	0377 Environment data string terminator	: D$RMS
abr	*EOF*	End Of File	
val	*$EOFGET*	000000 Get the current EOF LSN	: D$RMSIO
val	*$EOFSET*	000001 Set the current EOF LSN	: D$RMSIO
val	*$EOFWRIT*	2 Write to EOF immediately	: D$RMSIO
abr	*EOR*	End Of Record	
abr	*EOT*	End Of Transmission	
abr	*EOV*	End Of Volumn	
abr	*EPN*	Entry Point Names	
abr	*EPT*	Entry Point (Table)	
ufr	*$ERMSG*	Display RMS Minimum Error Message	: D$RMS
var	*$ERRC*	EXTERN $ERRC $ERRCODE; Standard RMS Err Code	:D$RMS
typ	*$ERRCODE*	STRUCT; RMS Standard Error Code	: D$RMS
sc	*$ERROR*	Abort a Program	: D$RMS
val	*$ES*	0231 End of string	: D$RMS
val	*$ESNF*	0271 End Of String, don't flush display	: D$RMSWS
sc	*$EXIT*	Exit a Program	: D$RMS
abr	*EXT*	Extension	
fld	*$EXT*	$EXTT; Extension	: $NAMEEXT
fld	*$EXT*	$EXTT; Extension	: $NAMEEXTENV
ddw	*EXTERN*	Declare a Name Defined in Another Program Module	
typ	*$EXTT*	[4] CHAR; File Extension	: D$RMS

val	FALSE	000000 Boolean value: 'false'	D$INC
abr	FAR	File Access Routine	
val	$FARIFVR	000004 FARCDEFS/SRC version	D$FAR
ddw	FAST	Future Code Generators; Var. Resides in Register	
abr	FAT	File Access Token	
abr	FAV	File Access Variable	
abr	FAVM	File Access Variable Marker	
abr	FBE	Free Buffer Enquires	
abr	FCB	File Control Block	
typ	$FCBA	STRUCT; File Control Block for AIM File	D$FAR
fld	$FCBACB	$ACB; Start of the ACB	$FCBA
typ	$FCBAIM	Macro to Configure AIM File Control Block	D$FAR
typ	$FCBAIMI	Initializer Macro used by $FCBAIM	D$FAR
fld	$FCBBLKL	BYTE; Number of AIM PFDB buffers	$FCBA
fld	$FCBBLKL	BYTE; Length of ISAM block in sectors	$FCBIS
typ	$FCBD	STRUCT; File Control Block for Direct File	D$FAR
fld	$FCBDBFS	BYTE; Number of bufrs associated with D.PFDB	$FCBA
fld	$FCBDBFS	BYTE; Number of bufrs associated w/D.PFDB	$FCBD
fld	$FCBDBFS	BYTE; Nbr of bufrs associated with D.PFDB	$FCBIS
fld	$FCBDCB	$DCB; Start of the DFCB (data file)	$FCBA
fld	$FCBDCB1	$DCB; Start of the DFCB (data file)	$FCBIS
fld	$FCBDCB2	$DCB; Start of the data control block	$FCBD
typ	$FCBDIR	Macro to Configure Direct File Control Block	D$FAR
typ	$FCBDIRPRT	Macro used by $FCBPRT, $FCBDIR, $FCBDOVR	D$FAR
typ	$FCBDIRPRTI	Initializer Macro used by $FCBDIRPRT	D$FAR
typ	$FCBDOVR	Macro to Configure Direct-Overlapped I/O FCB	D$FAR
fld	$FCBFLG1	BYTE; Flag byte 1	$FCBA
fld	$FCBFLG1	BYTE; Flag byte 1	$FCBD
fld	$FCBFLG1	BYTE; Flag byte 1	$FCBIS
fld	$FCBFLG2	BYTE; AIM flag byte	$FCBA
fld	$FCBFLG2	BYTE; User ISAM flag byte	$FCBIS
fld	$FCBHASH	BYTE; Hash code for data file name	$FCBA
fld	$FCBHASH	BYTE; Hash code for data file name	$FCBIS
fld	$FCBICB	$ICB; Start of the ICB	$FCBIS
typ	$FCBIS	STRUCT; File control block for ISAM file	D$FAR
typ	$FCBISAM	Macro to Configure ISAM File Control Block	D$FAR
typ	$FCBISAMI	Initializer Macro used by $FCBISAM	D$FAR
fld	$FCBKEY	^ ^ CHAR; Pointer to user's key list	$FCBA
fld	$FCBKEY	^ CHAR; Address of the user's ISAM key area	$FCBIS
fld	$FCBKLGT	BYTE; Number of keys in key list	$FCBA
fld	$FCBKLGT	BYTE; Length of the ISAM key	$FCBIS
fld	$FCBLINK	[2] UNSIGNED; Primary links	$FCBA
fld	$FCBLINK	[4] BYTE; Primary links	$FCBIS
typ	$FCBPRT	Macro to Configure Print File Control Block	D$FAR
fld	$FCBRLGT	UNSIGNED; Length of the user record	$FCBA
fld	$FCBRLGT	UNSIGNED; Length of the user record	$FCBD
fld	$FCBRLGT	UNSIGNED; Length of the user record	$FCBIS
fld	$FCBSLLH	[2] UNSIGNED; Secondary links	$FCBA
fld	$FCBSLLH	[4] BYTE; Secondary links	$FCBIS
fld	$FCBUREC	^ CHAR; User record address	$FCBA
fld	$FCBUREC	^ CHAR; User record address	$FCBD
fld	$FCBUREC	^ CHAR; User record address	$FCBIS
val	$FCSBIN	00004 Opened file is binary :$FCBA.$FCBFLG1	D$FAR
val	$FCSCMPR	00002 Compressed records :$FCBA.$FCBFLG1	D$FAR
val	$FCSIDUP	00001 Indicates that duplicate keys are allowed	

```
    $FCSIDUP                                   :$FCBIS.$FCBFLG2 : D$FAR
val $FCSISHR    00002 Shared use flag          :$FCBIS.$FCBFLG2 : D$FAR
val $FCSMNGD    00020 This file is managed at an FMT
    $FCSMNGD                                    :$FCBA.$FCBFLG1 : D$FAR
val $FCSNKEY    00004 Indicates key info not to be returned
    $FCSNKEY                                   :$FCBIS.$FCBFLG2 : D$FAR
val $FCSOPEN    00010 This file is open         :$FCBA.$FCBFLG1 : D$FAR
val $FCSOVER    00001 Overlapped I/O            :$FCBA.$FCBFLG1 : D$FAR
val $FCSTACB    00300 Primary AIM FCB           :$FCBA.$FCBFLG1 : D$FAR
val $FCSTDCB    0100 DFCB (direct or byte)      :$FCBA.$FCBFLG1 : D$FAR
val $FCSTICB    000040 Primary ISAM FCB         :$FCBA.$FCBFLG1 : D$FAR
val $FCSTMSK    000340 FCB type mask            :$FCBA.$FCBFLG1 : D$FAR
val $FCSTPRT    0200 Printer FCB - DISK res      :$FCBA.$FCBFLG1: D$FAR
val $FCSTPRU    0240 Printer FCB - PRINT res     :$FCBA.$FCBFLG1: D$FAR
val $FCSTSAB    0340 Secondary AIM FCB           :$FCBA.$FCBFLG1: D$FAR
val $FCSTSIB    0140 Secondary ISAM FCB          :$FCBA.$FCBFLG1: D$FAR
abr  FCT        File Creation Time
abr  FCV        File Control Variable
ufr $FDPACK     Numeric, Pack Two Decimal Numbers          : D$UFRNUM
abr  FDT        File Description Table
fld $FDTACCO    $ACCODES; Access Code                      : $FILEKEY
fld $FDTKEYL    $PACKPW; Packed Password                   : $FILEKEY
val $FDTKEYN    9  Number of keys                           : D$RMSIO
ufr $FDUNPAK    Unpack Character Into Two ASCII Digits     : D$UFRNUM
fld $FFLAG      BOOLEAN; File open flag                   : $FILESTBL
abr  FFMT       File Format
typ  FFMTABL$   EXTERN FFMTABL$ [0] STRUCT;File Fmt Table: D$UFRGEN
val $FFMTAIM    0020 AIM format                 :$OPENPT.$OTFMT : D$RMS
val $FFMTBAS    0007 Basic Object Code          :$OPENPT.$OTFMT : D$RMS
val $FFMTBIN    0013 Binary Data                :$OPENPT.$OTFMT : D$RMS
val $FFMTDBC    0006 Databus Object Code        :$OPENPT.$OTFMT : D$RMS
val $FFMTISM    0003 Isam Index                 :$OPENPT.$OTFMT : D$RMS
val $FFMTJOB    0012 CHAIN Job File             :$OPENPT.$OTFMT : D$RMS
val $FFMTL55    0004 Loadable 5500 REL/ABS Object Library
    $FFMTL55                                    :$OPENPT.$OTFMT : D$RMS
val $FFMTL66    0024 Loadable 6600/8x00 product     :$OPENPT : D$RMS |*
val $FFMTL80    0025 Loadable 8600/8800 product     :$OPENPT : D$RMS |*
val $FFMTMAC    0010 Macro Library              :$OPENPT.$OTFMT : D$RMS
val $FFMTMFD    0015 Managed File Descriptor:$OPENPT.$OTFMT : D$RMS
val $FFMTPTR    050  File Pointer file          :$OPENPT.$OTFMT : D$RMS |*
val $FFMTR55    040  Released 5500/3800 product   : $OPENPT : D$RMS |*
val $FFMTR66    041  Released 6600/8x00 product   : $OPENPT : D$RMS |*
val $FFMTR80    042  Released 8600/8800 product   : $OPENPT : D$RMS |*
val $FFMTRAC    0005 Non-Loadable REL/ABS Object Library
    $FFMTRAC                                    :$OPENPT.$OTFMT : D$RMS
val $FFMTRPM    0040 Min val, released product types:$OPENPT: D$RMS |*
val $FFMTSYS    0000 System File                :$OPENPT.$OTFMT : D$RMS
val $FFMTTMP    0001 Temporary File (Must be a 1)
    $FFMTTMP                                    :$OPENPT.$OTFMT : D$RMS
val $FFMTTXT    0002 Text (Logical Records) :$OPENPT.$OTFMT : D$RMS
val $FFMTUPF    0016 Universal Print Format :$OPENPT.$OTFMT : D$RMS
val $FFMTUTX    0014 Uncompressed Text Data :$OPENPT.$OTFMT : D$RMS
val $FFMTWPF    0017 Word Processing Format :$OPENPT.$OTFMT : D$RMS
val $FFMTWPS    0011 Word Processing Library:$OPENPT.$OTFMT : D$RMS
val $FFMTXFD    021  Extendd file (multi-volume file):$OPENPT:D$RMS |*
```

fld	***$FFTCODE***	BYTE;File Format Code	: FFMTABL$
fld	***$FFTNAME***	[4] CHAR; File Format Name	: FFMTABL$
abr	***FIFO***	First In, First Out	
fld	***$FILALLO***	$LSN;LSN of last allocated sector	: $FILEINFO
val	***$FILANYR***	010 Some Fld Entry is Reqd:$FILESPK	: D$RMS
val	***$FILECHK***	03 $FILES mode: Check for file opened	:D$RMSIO
val	***$FILEDAT***	4 Get FDT data for specific FAV	: D$RMSIO
ufr	***$FILEFMT***	Convert File Fmt Codes to ASCII String	: D$UFRGEN
val	***$FILEGET***	02 $FILES mode: Get file name from FAV	:D$RMSIO
typ	***$FILEINFO***	STRUCT;Info from $FILENAM Mode of $FILES	: D$RMSIO
typ	***$FILEKEY***	STRUCT;File Key Structure	: D$RMSIO
typ	***$FILEKEYS***	[$FDTKEYN] $FILEKEY; File Key List Array	: D$RMSIO
val	***$FILENAM***	1 $FILES mode:Get file names from FDT-PCN's	
ufr	***$$FILENAM***	Obtain the Next File Name in Catalog	: D$UFRENV
	$FILENAM		: D$RMSIO
val	***$FILENVR***	0004 Environment Required:$FILESPK	: D$RMS
val	***$FILEPCN***	00 $FILES mode: Get file FDT-PCN's	: D$RMSIO
ufr	***$$FILEPCN***	Open Catalog File and Obtain PCNs	: D$UFRENV
ufr	***$$FILEPCU***	Special Entry to $$FILEPCN	: D$UFRENV
typ	***$FILEPTR***	STRUCT; File Pointer Structure	: D$RMS
sc	***$FILES***	Multi-Resource, Obtain Disk File Info	: D$RMSIO
typ	***$FILESPK***	STRUCT; $SCANFLS File Specification	: D$RMS
typ	***$FILESTBL***	STRUCT; $FILES, $GETSFI Param Structure	: D$RMSIO
val	***$FILEXTR***	0002 Extension Required : $FILESPK	: D$RMS
fld	***$FILFCT***	$TIME; File creation time	: $FILEINFO
val	***$FILFDEF***	0020 The Field is Defined : $FILESPK	: D$RMS
fld	***$FILFEXT***	$EXTT; Extension	: $FILEINFO
fld	***$FILFFMT***	BYTE; File format code	: $FILEINFO
fld	***$FILFINC***	UNSIGNED; File increment in sectors	: $FILEINFO
fld	***$FILFLEN***	$LSN; LSN of EOF sector	: $FILEINFO
fld	***$FILFNAM***	$NAMET; File name	: $FILEINFO
val	***$FILNAMR***	0001 Name Required : $FILESPK.$FSOOPT	: D$RMS
val	***$FILNDSP***	0100 Inhibit Dspy if undef : $FILESPK	: D$RMS
val	***$FILQMOK***	040 Query/More/Not Marks Allwd : $FILESPK	: D$RMS
fld	***$FILSEGM***	BYTE; Number of used segments	: $FILEINFO
val	***$FINDNOD***	077 '?'find the node with the givn resource	:D$RMSIO
val	***$FMS***	000303 'FMS'	: D$ERRNUM
abr	***FMS***	File Management System	
val	***$FMT***	000304 'FMT'	: D$ERRNUM
abr	***FMT***	File Management Task	
fld	***$FNAMP***	^ $FILEINFO; File info storage pointer	: $FILESTBL
	$FNAMP	000010 Pointer to file name ($FILENAME)	
val	***$FOREVER***	000377	: D$RMSIO
sc	***$FORMAT***	Multi-Resource, Format a Unit on the Disk	: D$RMSIO
fld	***$FP***	LONG;	: $FILEPTR
fld	***$FPCN***	UNSIGNED; PCN of file ?	: $FILESTBL
fld	***$FPCNP***	^ UNSIGNED; Pointer to PCN's	: $FILESTBL
fld	***$FPTRBUFOF***	BYTE; Offset within sector	: $FILEPTR
fld	***$FPTRLSN***	$LSN; LSN	: $FILEPTR
ufr	***$FSCAN***	CmdInt, Compress a $FILESPK	: D$UFRSCAN
abr	***FSD***	File Structure Directory	
abr	***FSL***	File Security Level	
fld	***$FSODENV***	^ $ENVN; Default environment pointer	: $FILESPK
fld	***$FSODEXT***	^ $EXTT; Default extension pointer	: $FILESPK
fld	***$FSODNAM***	^ $NAMET; Default name pointer	: $FILESPK

fld	***$FSOOPT***	Option specification qualification	: $FILESPK	
fld	***$FSOSFT***	$SFENT; Symbolic file table	: $FILESPK	
fld	***$FTODO***	UNSIGNED; Number of PCN's to convert	: $FILESTBL	
fld	***$FUNC***	BYTE; Routine number	: $ERRCODE	
ddt	***"FUNCTION"***	DASL variable type; Indicated by (); the word ***"FUNCTION"*** is not a reserved word.		
fld	***$FX1***	[6] BYTE;	: $FILESTBL	
fld	***$FX2***	BYTE;	: $FILESTBL	
fld	***$FX3***	BYTE;	: $FILESTBL	
val	***$gBLCorn***	0004 Bottom left corner: └	: D$WORKSTN	|*
val	***$gBRCorn***	0005 Bottom right corner: ┘	: D$WORKSTN	|*
val	***$gBTBar***	0011 Bottom T-bar: ┴	: D$WORKSTN	|*
val	***$gCommand***	0022 Command: □	: D$WORKSTN	|*
val	***$gCROSS***	0014 Cross: ┼	: D$WORKSTN	|*
val	***$gCurs***	0000 Cursor: █	: D$WORKSTN	|*
val	***$gCursD***	0016 Cursor Down: ↓	: D$WORKSTN	|*
val	***$gCursL***	0017 Cursor Left: ←	: D$WORKSTN	|*
val	***$gCursR***	0020 Cursor Right: →	: D$WORKSTN	|*
val	***$gCursU***	0015 Cursor Up: ↑	: D$WORKSTN	|*
ufr	***$GDATTIM***	Obtain Current ASCII Date/Time String	: D$UFRGEN	
ufr	***$GENSMSK***	Generate Generic Scanning Masks	: D$UFRSCAN	
ufr	***$GENSTST***	Name Test-Under-Mask and Generate	: D$UFRSCAN	
val	***$gEnter***	0021 Enter: ↵	: D$WORKSTN	|*
ufr	***$GETCHN***	Get Response from CHAIN File / WS	: D$UFRWS	
ufr	***$GETCHTO***	Timeout for GETCSTK$ and GETCHN$	: D$UFRWS	
ufr	***$GETCSTK***	Get Response from Command Stack / WS	: D$UFRWS	
ufr	***$GETCSTO***	Timeout for GETCSTK$ and GETCHN$	: D$UFRWS	
sc	***$GETIME***	Obtain Current System Time	: D$RMSGEN	
ufr	***$GETLINE***	Get Response from Stack, CHAIN, or WS	: D$UFRWS	
ufr	***$GETLNTO***	Timeout Controlled Version of GETLINE	: D$UFRWS	
ufr	***$GETPASS***	Obtain, Compress Password from Keyin	: D$UFRENV	
sc	***$GETSFI***	Obtain Symbolic File Identification	: D$RMSIO	
val	***$gHBar***	0007 Horizontal bar: ─	: D$WORKSTN	|*
val	***$gLTBar***	0012 Left T-bar: ├	: D$WORKSTN	|*
sc	***$GLUTEN***	Get Last User Task Error Number	: D$RMSPROG	
abr	***GMT***	Greenwich Mean (Co-ordinated Universal) Time		
dcw	***GOTO***	Transfer Control to Labled Statement in Same Func		
val	***$gRTBar***	0013 Right T-bar: ┤	: D$WORKSTN	|*
fld	***$GSFIEXT***	$EXTT; File extension	: $SFITABLE	
fld	***$GSFIHSI***	$HSI; File HSI	: $SFITABLE	
fld	***$GSFINAM***	$NAMET; File name	: $SFITABLE	
fld	***$GSFINET***	$NAMET; Network containing the node	: $SFITABLE	
fld	***$GSFINOD***	$NAMET; Node containing the resource	: $SFITABLE	
fld	***$GSFIRES***	$NAMET; Resource containing the file	: $SFITABLE	
val	***$gTLCorn***	0002 Top left corner: ┌	: D$WORKSTN	|*
val	***$gTRCorn***	0003 Top right corner: ┐	: D$WORKSTN	|*
val	***$gTTBar***	0010 Top T-bar: ┬	: D$WORKSTN	|*
val	***$gUserMeta***	0177 User data meta character: ■	: D$WORKSTN	|*
val	***$gVBar***	0006 Vertical bar: │	: D$WORKSTN	|*
val	***$gVBBar***	0030 Vantage bottom bar: _	: D$WORKSTN	|*
val	***$gVBLCorn***	0025 Vantage bottom left corner: L	: D$WORKSTN	|*
val	***$gVBRCorn***	0026 Vantage bottom right corner: ┘	: D$WORKSTN	|*
val	***$gVCurs***	0001 Vantage cursor: ▯	: D$WORKSTN	|*

val	***$gVLBar***	0031 Vantage left bar: |	: D$WORKSTN |*
val	***$gVRBar***	0032 Vantage right bar: ▐	: D$WORKSTN |*
val	***$gVTBar***	0027 Vantage top bar: z	: D$WORKSTN |*
val	***$gVTLCorn***	0023 Vantage top left corner: ⌐	: D$WORKSTN |*
val	***$gVTRCorn***	0024 Vantage top right corner: ¬	: D$WORKSTN |*
val	***$H***	0234 New cursor column follows (pos)	: D$RMS
val	***$HA***	0236 Cursor column adjustment follows (adj)	:D$RMSWS
val	***$HD***	0242 Home down to lower left-hand corner	: D$RMS
abr	***HFD***	Hashed File Directory	
abr	***HSI***	Hierarchical Structure Information	
typ	***$HSI***	[32] CHAR; Hierarchical Structure Info Array	:D$RMS
val	***$HU***	0241 Home up to upper left-hand corner	: D$RMS
abr	***I/O***	Input / Output	
abr	***IAC***	Initial Access Code	
fld	***$ICACCNT***	[4] BYTE; Activity counter,read & write	:$INFOITEM
abr	***ICB***	ISAM Control Block	
typ	***$ICB***	STRUCT; ISAM Control Block	: D$FAR
fld	***$ICBACC***	BYTE; Index file access flags ($AC..)	: $ICB
fld	***$ICBCKEY***	^ CHAR; Address of the key save area	: $ICB
fld	***$ICBCURS***	^ BYTE; Current block and offset (LSB,MSB)	: $ICB
fld	***$ICBDCUR***	$FILEPTR; Data file cursor (LSB.MSB)	: $ICB
fld	***$ICBFLG1***	BYTE; Flag byte 2 (see $IFSTB LRIOCDEF/SRC)	: $ICB
fld	***$ICBLFP***	$LSN; LSN of the ISAM LFP (LSB.MSB)	: $ICB
fld	***$ICBMXKL***	BYTE; Maximum key length	: $ICB
fld	***$ICBPFDB***	$PFDB; Start of the index PFDB	: $ICB
fld	***$ICBTOP***	$LSN; LSN of top of tree (LSB.MSB)	: $ICB
fld	***$ICECNT1***	UNSIGNED; Error counter 1	: $INFOITEM
fld	***$ICECNT2***	UNSIGNED; Error counter 2	: $INFOITEM
fld	***$ICECNT3***	UNSIGNED; Error counter 3	: $INFOITEM
fld	***$ICECNT4***	UNSIGNED; Error counter 4	: $INFOITEM
fld	***$ICECNT5***	UNSIGNED; Error counter 5	: $INFOITEM
fld	***$ICID***	UNSIGNED; Controller variable identifier	:$INFOITEM
fld	***$ICKIND***	(...);Controller kind ($CK...)	: $INFOITEM
far	***$ICLOSE***	ISAM, Close	: D$FAR
fld	***$ICMBANK***	BYTE; Memory bank number of CTV	: $INFOITEM
fld	***$ICMBITS***	[22] BYTE; 22 bit error counters in CTV	: $INFOITEM
val	***$ICONTV***	022 Return all controller variables	: D$RMSGEN
fld	***$ICPORT***	BYTE; Logical port number	: $INFOITEM
far	***$IDEL***	ISAM Random, Delete	: D$FAR
far	***$IDELCR***	ISAM Seq, Delete Current Record Key, Data	: D$FAR
far	***$IDELK***	ISAM Random, Delete Record's Key	: D$FAR
abr	***IDENT***	Identification Sector	
val	***$IDLMTAB***	000020 Return delimiter table	: D$RMSGEN
abr	***IEOS***	Interated Electronic Office Station	
dcw	***IF***	Execute THEN Expression if Argument Non-zero, etc	
abr	***IFDB***	Index File Descriptor Block	
dcm	***IFELSE***	If First 2 Strings are Equal Result is 3rd,Else 4	
far	***$IINS***	ISAM Random, Insert	: D$FAR
far	***$IIOCLR***	ISAM, I/O Clear	: D$FAR
fld	***$ILFLAGS***	$NODEFLAGS; Flags ($INF...)	: $INFOITEM
fld	***$ILIGCNT***	UNSIGNED; Ignored received message count	:$INFOITEM

val	$ILINKAL	000015 Return all connection links	D$RMSGEN
val	$ILINKND	000014 Find named connection link	D$RMSGEN
fld	$ILNAME	$NAMET; Name of link and net linked to	$INFOITEM
typ	ILONG	STRUCT; 24 Bit Number Structure	D$INC
fld	$ILRCONF	UNSIGNED; Reconfiguration counter	$INFOITEM
fld	$ILRXMES	ULONG; Received message counter	$INFOITEM
fld	$ILTXABT	UNSIGNED; Xmission aborted (TA timeout)	$INFOITEM
fld	$ILTXERR	UNSIGNED; Transmissions with no TMA	$INFOITEM
fld	$ILTXMES	ULONG; Transmitted message counter	$INFOITEM
abr	IMOD	?	
val	$IMYNODE	017 Return name of local node	D$RMSGEN
fld	$INBCLSL	CHAR; Command/DLL library suffix letter	$INFOITEM
fld	$INBNETN	$NAMET; Boot net name	$INFOITEM
fld	$INBNLPT	BYTE; Nucleus library processor type	$INFOITEM
fld	$INBNLSL	CHAR; Nucleus library suffix letter	$INFOITEM
fld	$INBNODN	$NAMET; Boot node name	$INFOITEM
fld	$INBRESN	$NAMET; Boot resource name	$INFOITEM
dcm	INCLUDE	Obtain Program Input Lines from Specified File	
dcm	INCR	Produce a Value by Incrementing the Argument by 1	
abr	INDEP	Independent (Task)	
fld	$INDID	BYTE; Destination identification	$INFOITEM
fld	$INETNAM	$NAMET; Network name	$INFOITEM
val	$INFCFU	010 Checking file in use : $NODEFLAGS	D$RMSGEN
val	$INFCONG	004 ANV in process of connectn:$NODEFLAGS	D$RMSGEN
val	$INFFMA	040 FAV Markers Available : $NODEFLAGS	D$RMSGEN
val	$INFIFS	020 Incoming file support configured	
	$INFIFS	: $NODEFLAGS	D$RMSGEN
fld	$INFLAG	$NODEFLAGS; Flags ($INF...)	$INFOITEM
sc	$INFO	Obtain System Configuration Information	D$RMSGEN
fld	$INFODUMMY	[50] BYTE;	$INFOITEM
val	$INFOFF	001 Offline: ANV disconnect; CLV hard error	
	$INFOFF	: $NODEFLAGS	D$RMSGEN
typ	$INFOITEM	UNION; Info Returned by $INFO	D$RMSGEN
val	$INFTXER	002 Transmitter blocked by error	
	$INFTXER	: $NODEFLAGS	D$RMSGEN
val	$INODEAL	000013 Return all available nodes	D$RMSGEN
val	$INODEND	000012 Find named available node	D$RMSGEN
fld	$INONAME	$NAMET; Node name	$INFOITEM
fld	$INSTART	$TIME; Startup time	$INFOITEM
ddt	INT	DASL scalar data type: 2 bytes signed	
typ	$INTS	STRUCT; Interupt State Table	D$RMSPROG
fld	$INTSCC	BYTE; Condition code	$INTS
fld	$INTSRAD	^ BYTE; Return address	$INTS
fld	$INTSREG	[8] BYTE; Registers	$INTS
fld	$INTSXAD	$STARTADR; Execute address	$INTS
fld	$INVRP	[5] CHAR; Ver/Rev/Pre ASCII letters vvrrp	$INFOITEM
far	$IOPEN	ISAM, Open	D$FAR
abr	IPL	Initial Program Load	
far	$IPOS	ISAM Random, Position	D$FAR
far	$IPOSKP	ISAM Sequential, Position to Key Previous	D$FAR
far	$IPOSKS	ISAM Position to Next Key Sequential Record	D$FAR
far	$IPREP	ISAM, Prepare File	D$FAR
fld	$IRCDRVR	BYTE; Driver overlay ident. number	$INFOITEM
fld	$IRCOMET	BYTE; Error threshold	$INFOITEM
far	$IREAD	ISAM Random, Read	D$FAR

```
far $IREADCR   ISAM Sequential, Read Current                  : D$FAR
far $IREADKP   ISAM Sequential, Read Key Previous             : D$FAR
far $IREADKS   ISAM Sequential, Read Key Sequential           : D$FAR
val $IRF6      0100                        : $RSRCFLAGS : D$RMSGEN
val $IRFBSD    0200 Resource is byte string device
    $IRFBSD                                : $RSRCFLAGS : D$RMSGEN
val $IRFCHK    010  SYSCHECK in progress    :$RSRCFLAGS : D$RMSGEN
val $IRFOCP    002  Resource occupied      : $RSRCFLAGS : D$RMSGEN
val $IRFOFF    001  Resource offline       : $RSRCFLAGS : D$RMSGEN
val $IRFSPC    040  Special Open Mode - Open if Off-Line
    $IRFSPC                                : $RSRCFLAGS : D$RMSGEN
val $IRFSTP    020  Disk Sys Table problems  :$RSRCFLAGS: D$RMSGEN
val $IRFWRP    04   Resource write protected :$RSRCFLAGS: D$RMSGEN
val $IRMFAL    05   Return all multi file resources     : D$RMSGEN
val $IRMFND    04   Find named multi file resource      : D$RMSGEN
fld $IROCVID   UNSIGNED; Controller var.serial num.(") : $INFOITEM
fld $IROERCT   UNSIGNED; Error counter CARD READER     : $INFOITEM
fld $IROFCNT   UNSIGNED; Open files counter ( " )      : $INFOITEM
fld $IROFLAG   $RSRCFLAGS; Resource flags ( $IRF.. )   : $INFOITEM
fld $IROFREC   UNSIGNED; Free clusters (DISK ONLY)     : $INFOITEM
fld $IROHRDR   UNSIGNED; Hard read error counter       : $INFOITEM
fld $IROHRDW   UNSIGNED; Hard write error counter      : $INFOITEM
fld $IROKIND   BYTE; Resource kind ( $DK... )          : $INFOITEM
fld $IROMAXC   UNSIGNED; Max cluster avail ( " )       : $INFOITEM
fld $IRONAME   $NAMET; Source name (SPRM: net name ?)  : $INFOITEM
fld $IRORDCT   ULONG; Read activity counter            : $INFOITEM
fld $IROSCLU   BYTE; Sec. per cluster ( " )            : $INFOITEM
fld $IROSFTR   UNSIGNED; Soft read error counter       : $INFOITEM
fld $IROSFTW   UNSIGNED; Soft write error counter      : $INFOITEM
fld $IROSUBD   BYTE; Physical sub-device number        : $INFOITEM
fld $IROSUBK   BYTE; Resource sub-kind ( $SKDS..)      : $INFOITEM
fld $IROTIME   $TIME; Time of last access ( " )        : $INFOITEM
fld $IROWRCT   ULONG; Write activity counter           : $INFOITEM
val $IRSFAL    07 Return all single file resources      : D$RMSGEN
val $IRSFND    06 Find named single file resource       : D$RMSGEN
fld $IRUCFAV   BYTE; Number of consumed incoming FAVs  : $INFOITEM
val $IRUDATA   023 Return resource utilization data     : D$RMSGEN
fld $IRUFBUF   BYTE; Number of free incoming buffers   : $INFOITEM
fld $IRUFLAG   $IRUFLAGS; Flags (..$IRU.. )            : $INFOITEM
typ $IRUFLAGS  Resource Utilization Flags               : D$RMSGEN
val $IRUIFA    0010 Incomming file access supported
    $IRUIFA                                  : $IRUFLAGS : D$RMSGEN
val $IRUIFH    0002 Incoming filehandler in use
    $IRUIFH                                  : $IRUFLAGS  : D$RMSGEN
fld $IRUMBUF   BYTE; Peak value of ($IRUTBUF-$IRUFBUF) : $INFOITEM
fld $IRUMEMA   UNSIGNED; Nbr available memory sectors  : $INFOITEM
fld $IRUMFAV   BYTE; Peak value of $IRUCFAV            : $INFOITEM
fld $IRUMKBC   BYTE; Nmbr of bufrs used for FAV markers :$INFOITEM
fld $IRUMKFC   UNSIGNED; Number of free FAV markers    : $INFOITEM
val $IRUOFH    004 Outgoing filehandler in use:$IRUFLAGS :D$RMSGEN
fld $IRUOVAC   UNSIGNED; Overlay access counter        : $INFOITEM
fld $IRUOVLD   UNSIGNED; Overlay load counter          : $INFOITEM
fld $IRUOVWT   BYTE; ?                                 : $INFOITEM
fld $IRUSRWT   BYTE; ?                                 : $INFOITEM
fld $IRUSTEA   UNSIGNED; System table end address      : $INFOITEM
```

fld	***$IRUSTFA***	UNSIGNED; System table first address : $INFOITEM
fld	***$IRUTBUF***	BYTE; Total number of incoming buffers : $INFOITEM
fld	***$IRUTFAV***	BYTE; Total number of incoming FAVs : $INFOITEM
fld	***$IRUWFAV***	BYTE; Num.incoming FAVs waiting on bufrs :$INFOITEM
val	***$IRUx***	1 **UNUSED** was $IRUOVL : $IRUFLAGS : D$RMSGEN
val	***$IRUx***	1 (Pre-emptable overlay in use? was $IRUOVL)
far	***$IRWRT***	ISAM Random, Rewrite : D$FAR
far	***$IRWRTCR***	ISAM Sequential, Rewrite Current : D$FAR
abr	***ISAM***	Indexed Sequential Access Method
val	***$ISPLOCK***	0200,Shared program locked into memory
	$ISPLOCK	: $INFOITEM.$ISPSTAT : D$RMSGEN
val	***$ISPMEM***	037 Shared Program PSK count : $INFOITEM :D$RMSGEN
fld	***$ISPNAME***	$NAMET; ? : $INFOITEM : D$RMSGEN
fld	***$ISPSTAT***	BYTE; ? : $INFOITEM
fld	***$ISPUSER***	BYTE; ? : $INFOITEM
val	***$ISPVAL***	21 Return all shared program variable : D$RMSGEN
val	***$ISPVND***	20 Return named shared program variable : D$RMSGEN
val	***$ISTARTT***	000021 Return system startup time : D$RMSGEN
val	***$ITASKAL***	000011 Return all tasks : D$RMSGEN
val	***$ITASKME***	000016 Return caller's task info : D$RMSGEN
val	***$ITASKND***	000010 Find named task : D$RMSGEN
fld	***$ITOACTM***	BYTE; Actual number of memory sectors : $INFOITEM
fld	***$ITOFATH***	BYTE; Father task identification : $INFOITEM
fld	***$ITOID***	BYTE; Task identification : $INFOITEM
fld	***$ITOMAXM***	BYTE; ? : $INFOITEM
fld	***$ITOMINM***	BYTE; Minimum number of sectors : $INFOITEM
fld	***$ITONAME***	$NAMET; Task name : $INFOITEM
fld	***$ITOPRTY***	BYTE; Priority : $INFOITEM
abr	***ITT***	Invitations To Transmit
far	***$IWRITE***	ISAM Random, Write : D$FAR
abr	***KDF***	Keyword Definitions File
abr	***KDS***	? (Serial Printer attached to 8600 KDS Port)
ufr	***$KEYCHAR***	Obtain One Translated Character : D$UFRWS
ufr	***$KEYCLR***	Clear the Keyin FIFO : D$UFRWS
ufr	***$KEYIN***	Accept a String From Keyboard to Memory : D$UFRWS
ufr	***$KEYINTO***	Timeout Controlled Version of KEYIN$: D$UFRWS
var	***KEYSECS$***	EXTERN KEYSECS$ BYTE; Timeout for $KEYIN : D$UFRWS
abr	***KTT***	Keyboard Translate Table
abr	***KWIC***	Keyword in Context
fld	***$LACODE***	[256] BYTE; absolute code : $LIBSECTOR
fld	***$LAHDR***	$ABSHDR; absolute header : $LIBSECTOR
ufr	***$LBADD***	Add a Member to a Library : D$UFRLIB
ufr	***$LBDEL***	Delete a Member From a Library : D$UFRLIB
ufr	***$LBFIND***	Locate Library Member : D$UFRLIB
ufr	***$LBFREE***	Find the First Free Sector in a Library : D$UFRLIB
ufr	***$LBGTLSN***	Locate Library Member and Return LSN : D$RMS
abr	***LCD***	Liquid Crystal Display |*
val	***$LDEL***	000377 Deleted data mark : D$RMS
val	***$LEOB***	000375 End of block mark : D$RMS
val	***$LEOF***	000373 End of file mark : D$RMS
val	***$LEOR***	000372 End of record mark : D$RMS

abr	*LFDB*	Logical File Descriptor Block		
val	*$LFLOKSP*	000 Lock specified FAV	: D$RMSSPEC	
abr	*LFP*	Logical File Pointer		
abr	*LFST*	Logical File State Table		
val	*$LFULOKS*	001 Unlock specified FAV	: D$RMSSPEC	
abr	*LGT*	Length		
val	*$LIBABSO*	04 "ABS" format overlay: $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBABSX*	03 "ABS" format executable:$LIBTYPE	:D$RMSSTRUCT	
fld	*$LIBDIR*	[16] $LIBENTRY; Library directory	: $LIBSECTOR	
val	*$LIBDLL*	014 ARC dwn-line load frmt:$LIBTYPE	:D$RMSSTRUCT	
typ	*$LIBENTRY*	STRUCT; Library Directory Entry	: D$RMSSTRUCT	
val	*$LIBEPN*	0007 Entry point names : $LIBTYPE	: D$RMSSTRUCT	
fld	*$LIBEPT*	UNSIGNED; Entry Point Address	: $ABSHDR	
fld	*$LIBFPAG*	UNSIGNED; First page	: $ABSHDR	
val	*$LIBFREE*	0000 Free entry : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBLINK*	2 Link to next directory sector		
	$LIBLINK	: $LIBTYPE	: D$RMSSTRUCT	
fld	*$LIBMAP*	[64] BYTE; Primary MAP First byte	: $ABSHDR	
fld	*$LIBMLEN*	UNSIGNED; Length in sectors	: $LIBENTRY	
fld	*$LIBMLEN*	UNSIGNED; Length in sectors	: $MEMBER	
fld	*$LIBMLMD*	UNSIGNED; Last modification date	: $LIBENTRY	
fld	*$LIBMLMD*	UNSIGNED; Last modification date	: $MEMBER	
fld	*$LIBMLSN*	UNSIGNED; First LSN	: $LIBENTRY	
fld	*$LIBMLSN*	UNSIGNED; First LSN	: $MEMBER	
fld	*$LIBMNAM*	$LNAMET; Member name	: $LIBENTRY	
fld	*$LIBMNAM*	$LNAMET; Member name	: $MEMBER	
val	*$LIBMT*	0 Memory sector empty (not used)		
	$LIBMT	: $ABSHDR	: D$RMSSTRUCT	
fld	*$LIBMTYP*	$LIBTYPE; Type of this member:	: $LIBENTRY	
fld	*$LIBMTYP*	BYTE; Type of this member ($LIBTYPE ?)	: $MEMBER	
fld	*$LIBMVRN*	BYTE; 3-bits version, 5-bits revision	: $LIBENTRY	
fld	*$LIBMVRN*	BYTE; 3-bits version, 5-bits revision	: $MEMBER	
val	*$LIBMXPG*	57 $LIBPG1 Array Size, maximum nbr of page groups		|*
	$LIBMXPG	: $ABSHDR	: D$RMSSTRUCT	
fld	*$LIBNAA*	UNSIGNED; Next Available Address	: $ABSHDR	
fld	*$LIBNOPG*	BYTE; Number of Page Groups	: $ABSHDR	
fld	*$LIBNPAG*	BYTE; Number of pages	: $ABSHDR	
fld	*$LIBPG1*	[$LIBMXPG] Page Groups	: $ABSHDR	
val	*$LIBPRIV*	1 Memory sector private: $ABSHDR	: D$RMSSTRUCT	
val	*$LIBRELL*	5 "REL" format : $LIBTYPE	: D$RMSSTRUCT	
fld	*$LIBSCODE*	$RELCODE; relocatable sector type	: $LIBSECTOR	
typ	*$LIBSECTOR*	UNION; Library Sector Formats	: D$RMSSTRUCT	
val	*$LIBSHAR*	2 Memory sector shared : $ABSHDR	: D$RMSSTRUCT	
fld	*$LIBSPID*	$NAMET: Shared Program I.D.	: $ABSHDR	|*
val	*$LIBT006*	6 : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBT008*	8 : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBT009*	9 : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBT010*	10 : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBT011*	11 : $LIBTYPE	: D$RMSSTRUCT	
val	*$LIBTBAD*	3 Illegal mem sector type : $ABSHDR	: D$RMSSTRUCT	
val	*$LIBTERM*	0001 End of library : $LIBTYPE	: D$RMSSTRUCT	
typ	*$LIBTYPE*	ENUM Lib Directory Entry Types	: D$RMSSTRUCT	
fld	*$LIBUCS*	UNSIGNED; Two Bytes Used;	: $ABSHDR	
fld	*$LIBXX*	[2] BYTE; ?	: $ABSHDR	
abr	*LIFO*	Last In, First Out		

fld	LINK	^ $ENVT; Link to next entry	: $ENVT	
val	$LMCV	DEFINE'd 0371 Minimum control character value	:D$RMS	
abr	LMT	? DEF ECLPS,2,'Release LMT'		
	LMT	? DEF ECLPS,8,'Patching -- LMT Error'		
typ	$LNAMET	[8] CHAR; Library Member Name	: D$RMS	
fld	$LNK1	UNSIGNED;	: $MAB	
fld	$LNK2	UNSIGNED;	: $MAB	
sc	$LOAD	Load an Overlay	: D$RMS	
ufr	$LOADREL	Invoking the Relocating Loader	: D$UFRRLD	
sc	$LOCKFAV	Lock/Unlock specified FAV	: D$RMSSPEC	|*
ufr	$LOCKRIM	RIM Lockout, Attempt to Open Pipe	: D$UFRSYS	
ufr	$LOGCLR	Clear Logging Flags	: D$UFRWS	
ufr	$LOGGING	Determine if Logging is Active	: D$UFRWS	
ufr	$LOGSET	Set Logging Flags	: D$UFRWS	
ddt	LONG	DASL scalar data type: 4 bytes signed		
dcw	LOOP	Execute Substatements Until WHILE Expression = 0		
abr	LOST	Locked Out Sector Table (actually CLUSTERS)		|*
abr	LPS	Large Program Support		
abr	LR	Logical Record		
abr	LRIO	Logical Record Input/Output		
fld	$LRLINE	$RELLINE; Rel DEBUG line numbers sector	:$LIBSECTOR	
abr	LRN	Logical Record Number		
fld	$LROBJ	$RELOBJ; Rel object code sector	: $LIBSECTOR	
fld	$LRPID	$RELPID; Rel program ID sector	: $LIBSECTOR	
fld	$LRXDEF	$RELXDEF;Rel external definition sector	:$LIBSECTOR	
fld	$LRXEPN	$RELEPNS; Rel entry point member sector	:$LIBSECTOR	
fld	$LRXFER	$RELXFER; Rel starting address sector	:$LIBSECTOR	
fld	$LRXREF	$RELXREF; Rel external reference sector	:$LIBSECTOR	
abr	LSB	Least Significant Byte		
abr	LSN	Logical Sector Number		
typ	$LSN	ULONG; Logical Sector Number	: D$RMS	
val	$LSPC	0371 Space compression count follows	: D$RMS	
val	$LST	0374 Special text mark	: D$RMS	
abr	LSW	Least Significant Word (16 bits)		
fld	LSW	UNSIGNED; Least significant 16 bits	: ILONG	
fld	LSW	UNSIGNED;	: ULONG	
abr	LT	Logical Tape		
val	$LXX	0376	: D$RMS	
abr	MAB	Managed Access Block		
typ	$MAB	STRUCT; Managed File Access Block	: D$FAR	
fld	$MABID	UNSIGNED; ID of entity being managed	: $MAB	
fld	$MABLGON	UNSIGNED; Log-on pipe FAV	: $MAB	
fld	$MABLINK	STRUCT, Links	: $MAB	
fld	$MABPRIV	UNSIGNED; Private pipe (request+response)FAV	: $MAB	
fld	$MABRQIN	UNSIGNED; Request init pipe FAV	: $MAB	
fld	$MABUAT	UNSIGNED; User access token (UAT) ID	: $MAB	
ufr	$MAP4K	System-IF, Allocate Memory For a PFDB	: D$UFRSYS	
val	MAXINT	077777 ,DEFINE'd	: D$INC	
val	MAXLONG	017777777777, DEFINE'd	: D$INC	
val	$MAXNPW	DEFINE'd 20 Max # of passwords in an env	: D$RMS	
val	$MAXPRBF	02 Max number of buffers allowed	: D$FAR	
val	MAXUNSIGNED	0177777, DEFINE'd	: D$INC	
abr	MB	Mega-Byte		

val	*$MCDSOFF*	4 Switch user task to single sector mode	: D$RMSMEM
val	*$MCDSON*	3 Switch user task to dual sector mode	: D$RMSMEM
val	*$MCDSTST*	5 Test sector table mode	: D$RMSMEM
val	*$MCMDOFF*	1 De-activate memory diagnostic task	: D$RMSMEM
val	*$MCMDON*	0 Activate memory diagnostic task	: D$RMSMEM
val	*$MCMDTST*	2 Perform memory diagnostic	: D$RMSMEM
abr	*MEM*	Memory	
typ	*$MEMBER*	STRUCT; Library Member Structure	: D$UFRLIB
sc	*$MEMCTL*	Memory Control	: D$RMSMEM
sc	*$MEMGET*	Obtain a Private Memory Sector	: D$RMSMEM
ufr	*$$MEMGET*	Obtain A Physical Memory Sector	: D$UFRSYS
sc	*$MEMKEY*	Obtain a Memory Sector Key	: D$RMSMEM
sc	*$MEMMAP*	Map a Memory Sector into Logical Space	: D$RMSMEM
sc	*$MEMPROT*	Change Memory Protection	: D$RMSMEM
sc	*$MEMREL*	Release a Memory Sector	: D$RMSMEM
ufr	*$$MEMREL*	Release a Physical Memory Sector	: D$UFRSYS
abr	*MFD*	Managed File Descriptor	
cr	*MFREMEM$*	Deallocate a Block of Memory	
cr	*MGETCLR$*	Allocate a Cleared Block of Memory	
cr	*MGETFST$*	Allocate Free Space Block of Memory	
cr	*MGETMEM$*	Allocate a Block of Memory	
cr	*MGETPAG$*	Obtain a Page of Logical Memory Space	
fld	*MILLISECONDS*	BYTE; Eight millisecond counter ??	: $SYSTIME
ufr	*$MLTPLY3*	Numeric, Unsigned 24-bit Multiplication	: D$UFRNUM
ufr	*$MMFREMEM*	Deallocate a Block of Memory	: D$UFRMEM
ufr	*$MMGETCLR*	Allocate a Cleared Block of Memory	: D$UFRMEM
ufr	*$MMGETFST*	Allocate Free Space Block of Memory	: D$UFRMEM
ufr	*$MMGETMEM*	Allocate a Block of Memory	: D$UFRMEM
ufr	*$MMGETPAG*	Obtain a Page of Logical Memory Space	: D$UFRMEM
cr	*MMGINIT$*	Initialize Memory Management	
ufr	*$MMINIT*	Initialize Memory Management	: D$UFRMEM
ufr	*$MMRETPAG*	Release Page of Logical Memory Space	: D$UFRMEM
abr	*MPCA*	Multi-Port Communications Adapter	
val	*$MPROTRO*	0200 Set memory to read only	: D$RMSMEM
val	*$MPROTRW*	0 Set memory to read/write	: D$RMSMEM
cr	*MRETPAG$*	Release a Page of Logical Memory Space	
fld	*MSB*	BYTE; Most siginificant byte	: ILONG
fld	*MSB*	BYTE;	: ULONG
abr	*MSB*	Most Significant Byte	
var	*$MSG*	EXTERN $MSG [$MSGLGT] CHAR; Bfr for $PUT UFRs:D$RMS	
ufr	*$MSGC*	Err-Msg, Locate a Msg and Copy Into $MSG	: D$RMS
ufr	*$MSGCGET*	Err-Msg, Locate and Deliver a Message	: D$UFRERR
ufr	*$MSGCXO*	Err-Msg, Open and Pos the Command Lib	: D$UFRERR
val	*$MSGLGT*	81 Length of $MSG buffer	: D$RMS |*
abr	*MSN*	Mapped Sector Number; index of PSKs in sector table	
	MSN	Each task has a sector table listing 16 PSKs	
	MSN	MSN 0 initially always contains the PSK of the PCR	
abr	*MSW*	Most Significant Word (16 bits?)	
abr	*MTX*	?	

abr	NAK	Negative Acknowledgement	
fld	NAME	$ENVN: Environment name	$ENVT
fld	$NAME	$NAMET; Name	$NAMEEXT
fld	$NAME	$NAMET; Name	$NAMEEXTENV
typ	$NAMEEXT	STRUCT; Name, and Extension	D$RMS
typ	$NAMEEXTENV	STRUCT; Name, Extension, and Environment	D$RMS
typ	$NAMET	[12] CHAR; File Name Type	D$RMS
abr	NBR	Number	
val	$NEWFILE	0001 : $ACCODES	D$RMSIO
ufr	$NEWPCR	Create New PCR for Independent Task	D$UFRENV
abr	NID	Next ID	
val	NIL	0 DEFINE'd	D$INC
val	$NL	0243 Advance to new line	D$RMSWS
val	$NO	000116 'N'	D$PCR
val	$NOADR	0177777 Indicate "NO ADDRESS GIVEN"	D$RMS
typ	$NODEFLAGS	Node Flags in $INFO	D$RMSGEN
val	$NOPSK	0377 DEFINE'd Indicate "No PSK given"	D$RMS
val	$NQDQ	0307 $NQDQ error class	D$ERRNUM
ufr	$NQDQBLD	Build a Message Block	D$UFRNQDQ
ufr	$NQDQCHK	Check Limited Request	D$UFRNQDQ
fld	$NQDQCNT	BYTE; Item count	$NQDQMSG
ufr	$NQDQENL	Request a Limited Enqueue	D$UFRNQDQ
ufr	$NQDQENQ	Enqueue on a Resource	D$UFRNQDQ
typ	$NQDQITEM	[6] BYTE; NQDQ List Item	D$UFRNQDQ
ufr	$NQDQLGF	Disconnect from the System	D$UFRNQDQ
ufr	$NQDQLGN	Connect to the System	D$UFRNQDQ
fld	$NQDQLIST	[0] $NQDQITEM; Item list	$NQDQMSG
typ	$NQDQMSG	STRUCT; NQDQ Message	D$UFRNQDQ
ufr	$NQDQREL	Release a Resource	D$UFRNQDQ
ufr	$NQDQRST	Reset the Controller 4-byte Counters	D$UFRNQDQ
ufr	$NQDQSTA	Acquire Controller Statistics	D$UFRNQDQ
typ	$NQDQSTAT	STRUCT; NQDQ Statistics	D$UFRNQDQ
ufr	$NQDQSTP	Terminate an In-Prog Limited Enqueue	D$UFRNQDQ
ufr	$NQDQWAT	Wait for a Limited Enqueue Request	D$UFRNQDQ
val	$NRPRIOR	8 Number of priority levels	D$RMSPROG
val	$NS	0233 New string address follows ((loc))	D$RMSWS

fld	$O	$OPENPT;	$OPENPTS
abr	OBJ	Object File Output or Input (compilers)	
val	$OMBYPAS	007 Open mode: Bypass passwrd/security checks	D$RMS
val	$OMCHECK	005 Open mode: Disk structure check access	D$RMS
val	$OMCREAT	004 Open mode: Create a new file	D$RMS
val	$OMEXCL	002 Open mode: Exclusive read/write access	D$RMS
val	$OMPREP	003 Open mode: Open or create file	D$RMS
val	$OMREAD	000 Open mode: Shared read-only access	D$RMS
val	$OMREPAR	006 Open mode: Disk structure repair access	D$RMS
val	$OMSHARE	001 Open mode: Shared read/write access	D$RMS
ufr	$OPEN	Get OPENPT Link Set to Env Data Area	D$RMS
sc	$OPENENV	Open a File	D$RMSIO
typ	$OPENPT	STRUCT; Open Parameter Table	D$RMS
typ	$OPENPTS	UNION; $OPENENV in $OMCHECK,$OMREPAR Modes	D$RMSIO
val	$OPTFDEF	01 Returned if option given : $OPTION	D$RMS
fld	$OPTFLG	Option flag bits	$OPTION
val	$OPTFQOK	004 Option value may be quotd:$OPTION.$OPTFLG	D$RMS

val	***$OPTFVAL***	2 Returned if value given	: $OPTION : D$RMS
typ	***$OPTION***	STRUCT; $SCANFOS Option Specification	: D$RMS
fld	***$OPTMAX***	BYTE; Option value maximum length	: $OPTION
fld	***$OPTSON***	[$SONT] CHAR; Option field name	: $OPTION
fld	***$OPTSTR***	[0] CHAR; Option value string	: $OPTION
typ	***$OPTTAIL***	STRUCT; $SCANOS Option List Terminator	: D$RMS
val	***$OPTTERM***	0377, DEFINE'd	: D$RMS
fld	***$$OPTTERM***	BYTE; $OPTTERM (not $$) is defined 0377	: $OPTTAIL
fld	***$OPTTOT***	BYTE; Option global total count	: $OPTTAIL
fld	***$OPTVAL***	BYTE; Option value length	: $OPTION
val	***$OPTVCLR***	0377 Option not given	: $OPTION : D$RMS
val	***$OPTVSET***	0376 Option had no value	: $OPTION : D$RMS
fld	***$OTBLKL***	UNSIGNED; O,Maximum block length	: $OPENPTS
fld	***$OTCODE***	BYTE; O,Access code and file created flag	:$OPENPT
fld	***$OTENV***	^ CHAR; Pointer to ENV data name or entry	:$OPENPT
fld	***$OTFEOFB***	BYTE; O,End-of-File byte pointer	:$OPENPT
fld	***$OTFID***	UNSIGNED; O,File identification FDT-PCN	:$OPENPT
fld	***$OTFILE***	^ $NAMEEXT; I,Pointer to file name and ext	:$OPENPT
fld	***$OTFINC***	UNSIGNED; B,Space increment in sectors,	:$OPENPT
fld	***$OTFLEN***	$LSN; B,End-of-File location LSN,	:$OPENPT
fld	***$OTFMT***	ENUM(...); B,Format ($FFMT...)	:$OPENPT
fld	***$OTHFDLGT***	UNSIGNED; Nbr of hash file directory secs	:$OPENPTS
fld	***$OTHFDLSN***	BYTE; LSN of first HFD sector	: $OPENPTS
fld	***$OTIDLSN***	BYTE; O,Disk LSN of identification sector	:$OPENPTS
fld	***$OTKIND***	ENUM(...); O,Resource Kind ($DK...)	: $OPENPT
fld	***$OTNID***	UNSIGNED; ?	: $OPENPT
fld	***$OTONOS***	BYTE; O,Optimum number of sectors to do	: $OPENPT
fld	***$OTPFDB***	^ $PFDB; I,Pointer to PFDB,	: $OPENPT
fld	***$OTRID***	BYTE; O,Resource ID ARV serial number	: $OPENPT
fld	***$OTRTRY***	BYTE; I,Maximum number of retries	: $OPENPTS
fld	***$OTSQL***	BYTE; B,Security level	: $OPENPT
fld	***$OTSUBK***	BYTE; O,Resource sub-kind ($SK...)	: $OPENPT
fld	***$OTTIME***	$TIME; O,Time of creation in binary	: $OPENPT
fld	***$OTX1***	[9] BYTE;	: $OPENPTS
fld	***$OTX2***	[8] BYTE;	: $OPENPTS

abr	***PAB***	Program Address Blocks	
val	***$PABASS***	0200 PAB Assigned	: $PABFLAGS : D$RMSSTRUCT
typ	***$PABENTRY***	STRUCT; Rel. Prgm ID Sector PAB Entry	: D$RMSSTRUCT
val	***$PABF000***	0001 Unassigned	: $PABFLAGS : D$RMSSTRUCT
val	***$PABF001***	0002 Unassigned	: $PABFLAGS : D$RMSSTRUCT
val	***$PABFCOMN***	0010 Common PAB	: $PABFLAGS : D$RMSSTRUCT
val	***$PABFDATA***	004 Data PAB	: $PABFLAGS : D$RMSSTRUCT |*
typ	***$PABFLAGS***	Relocatable Prgm ID Sector PAB Flags	: D$RMSSTRUCT
val	***$PABFPS***	0020 PAB must not cross page boundry	
	$PABFPS		: $PABFLAGS : D$RMSSTRUCT
val	***$PABFREL***	0100 PAB is Relocatable	: $PABFLAGS : D$RMSSTRUCT
val	***$PABFTP***	0040 PAB must start on page boundry	
	$PABFTP		: $PABFLAGS : D$RMSSTRUCT
fld	***$PABTBADR***	UNSIGNED; PAB address	: $PABENTRY
fld	***$PABTBFLG***	$PABFLAGS; PAB Flags ($PAB...)	: $PABENTRY
fld	***$PABTBLGT***	UNSIGNED; PAB length	: $PABENTRY
fld	***$PABTBNAM***	[8] CHAR; PAB name	: $PABENTRY
abr	***PAC***	Packets	

typ	***$PACKPW***	[6] BYTE; Packed Password	: D$RMS	
fld	***PAGE***	BYTE; page number	: $PFDBBUF	
ufr	***$PAKPW***	Pack ACSII String Into Password	: D$UFRENV	
fld	***$PBLKL***	UNSIGNED; Block length in bytes:PIPE,TAPE	: $PFDB	
fld	***$PBUFL***	[1] $PFDBBUF; Start of buffer list	: $PFDB	
val	***$PCFREAD***	000000 Read a block of cards	: D$RMSIO	
fld	***$PCLSN***	$LSN; Logical sector number:DISK only	: $PFDB	
abr	***PCN***	Physical Cluster Number ?		
abr	***PCR***	Program Communication Region		
typ	***(PCR)***	EXTERNALLY defined structure of variables	: D$PCR	
	(PCR)			
	(PCR)	/* See TYPE Section PCR for all $PCR... stuff */		
	(PCR)			
var	***$PCR1STP***	∧ BYTE; Adr of 1st byte of PCR fixd data area	:D$PCR	
var	***$PCRABTF***	Abort reason flags : PCR	: D$RMS	
val	***$PCRAFGA***	000001 General abort : PCR	: D$RMS	
var	***$PCRBJCE***	$ENVN; BJF "WAIT" module extension	: D$PCR	
var	***$PCRBJCM***	$LNAMET; BJF "WAIT" module member name	: D$PCR	
var	***$PCRBJCN***	$NAMEEXT; BJF "WAIT" module name/ext	: D$PCR	
val	***$PCRCFAC***	000001 Chaining active	: D$PCR	
val	***$PCRCFCR***	000002 Next command line ready	: D$PCR	
val	***$PCRCFFO***	000004 Chain file open	: D$PCR	
val	***$PCRCFNA***	010 "No Abort" on $ERROR exit : $PCRCHNF	: D$PCR	
val	***$PCRCFRS***	020 CHAIN File has been Restarted :$PCRCHNF	:D$PCR	
var	***$PCRCHND***	∧ $NAMEEXTENV; Ptr to CHAIN cmd fname/ext:env	:D$PCR	
var	***$PCRCHNF***	Chaining flags	: D$PCR	
var	***$PCRCHNV***	[3] BYTE; Adr of entry vector; CHAIN routine	:D$PCR	
var	***$PCRCIF2***	SETW(): Second Command Interpreter Flags	: D$PCR	|*
var	***$PCRCLEL***	∧ CHAR; Command line syntax error location	: D$PCR	
var	***$PCRCMDE***	$ENVN; Env'ment name of command executing	: D$RMS	
var	***$PCRCMDF***	Command interpreter flags	: D$PCR	
var	***$PCRCMDM***	$LNAMET; Member Name being executed,or spaces	:D$PCR	
var	***$PCRCMDN***	$NAMEEXT; Filename/ext of comd executing	: D$RMS	
var	***$PCRCPFL***	BYTE; Current printer form length	: D$PCR	
var	***$PCRCWHL***	BYTE; Current Window Left Horizontal	: D$PCR	
var	***$PCRCWHR***	BYTE; Current Window Right Horizontal	: D$PCR	
var	***$PCRCWVL***	BYTE; Current Window Lower Vertical	: D$PCR	
var	***$PCRCWVU***	BYTE; Current Window Upper Vertical	: D$PCR	
val	***$PCRDF2F***	0100000 Secondary CI flags available		
	$PCRDF2F	: $PCRCMDF	: D$PCR	
val	***$PCRDFBJ***	000400 Batched job facility is active	: D$PCR	
val	***$PCRDFCF***	000040 Reset char font and translate table	: D$PCR	
val	***$PCRDFCW***	020000 Current Window data valid :$PCRCMDF	: D$PCR	
val	***$PCRDFEH***	000002 Display all heading records	: D$PCR	
val	***$PCRDFFK***	000020 Abort GETLINE$, KEYIN$ on function key	:D$PCR	
val	***$PCRDFMC***	04000 CMD Line was Menu-Generated :$PCRCMDF	:D$PCR	
val	***$PCRDFML***	040000 Menu line exists;		
	$PCRDFML	$PCRCLEL points to New Line : $PCRCMDF	: D$PCR	
val	***$PCRDFNC***	0100 No "STOP" bar;clears help wndow:$PCRCMDF	:D$PCR	|*
val	***$PCRDFNH***	000001 Inhibit signon/heading display	: D$PCR	
val	***$PCRDFNS***	010000 No STOP bar - does not clear window		|*
	$PCRDFNS	: $PCRCMDF	: D$PCR	
val	***$PCRDFNW***	000010 No workstation available	: D$PCR	
val	***$PCRDFSO***	000004 Cmd int entered from signon program	: D$PCR	
val	***$PCRDFWI***	02000 Standard window is active : $PCRCMDF	: D$PCR	

val	*$PCRDFWW*	001000 This is version II command int	: D$PCR
val	*$PCRDFWW*	01000 Standard window was active : $PCRCMDF	: D$PCR
var	*$PCREEXT*	*** File extension *** NOT IN DASL	
val	*$PCREFBC*	000001 $ERRC (BC) error code exists	: D$PCR
val	*$PCREFCI*	040 Error Msg is in C.I Msg Member :$PCRERRF	: D$PCR
val	*$PCREFFF*	000004 File in error flag	: D$PCR
val	*$PCREFNN*	000010 Net/node info present	: D$PCR
val	*$PCREFRF*	000002 Recursion flag	: D$PCR
val	*$PCREFTS*	020 Error Message is on top of Command Stack	
	$PCREFTS	: $PCRERRF	: D$PCR
var	*$PCREHSI*	*** File HSI *** NOT IN DASL	
var	*$PCRENAM*	*** File name *** NOT IN DASL	
var	*$PCRENET*	*** Net containing node *** NOT IN DASL	
var	*$PCRENOD*	*** Node containing resource *** NOT IN DASL	
var	*$PCRERES*	*** Resource containing file *** NOT IN DASL	
var	*$PCRERRF*	Command Int $ERROR reason flags	: D$PCR
val	*$PCRF2AW*	1 Task Running under Attached W/S Control	
	$PCRF2AW	: $PCRCIF2	: D$PCR
var	*$PCRFIEI*	[0] BYTE; $SFITABLE Start of area	: D$PCR
var	*$PCRFOF*	[0] BYTE; First byte of fixed PCR area	: D$PCR
var	*$PCRFVCI*	UNSIGNED; Locked FAV fro Command Interpreter	: D$PCR
var	*$PCRFVUP*	UNSIGNED; Locked FAV for User Program	: D$PCR
var	*$PCRKXT*	[128] CHAR; Keybd translate table; this WS	: D$PCR
val	*$PCRLFAC*	000001 Logging active	: D$PCR
val	*$PCRLFEO*	000004 Log only error messages	: D$PCR
val	*$PCRLFFO*	000020 Log file open	: D$PCR
val	*$PCRLFHR*	000040 HD/RU before each logged message	: D$PCR
val	*$PCRLFNI*	000010 Log note display inhibited	: D$PCR
val	*$PCRLFSP*	000002 Logging suspended	: D$PCR
var	*$PCRLOGD*	^ $NAMEEXTENV; Pntr to LOG Cmd fname/ext:env	: D$PCR
var	*$PCRLOGF*	Logging flag	: D$PCR
var	*$PCRLOGV*	[3] BYTE; Addr of entry vector; LOG routine	: D$PCR
var	*$PCRMIND*	[0] BYTE; Default bottom UET/CMD stack addr	: D$PCR
var	*$PCRMINP*	^ BYTE; Minimum usable PCR location	: D$PCR
var	*$PCRMLGF*	UNSIGNED; "LOGOFF"	: D$PCR
var	*$PCRMLGN*	UNSIGNED; "LOGON"	: D$PCR
var	*$PCRMLRC*	UNSIGNED; Check bits for Control Area	: D$PCR
var	*$PCRMMAL*	UNSIGNED; "MAIL"(optional)	: D$PCR
var	*$PCRMMSG*	UNSIGNED; "MESSAGE" member for Cmd Int	: D$PCR
var	*$PCRMNXL*	UNSIGNED; "NEXTLINE"	: D$PCR
var	*$PCRMRET*	UNSIGNED; "RETURN"	: D$PCR
var	*$PCRMRNL*	UNSIGNED; "RUNLINE"	: D$PCR
var	*$PCRNO*	CHAR; DEFINE($NO, 'N')	: D$PCR
var	*$PCRPSKA*	BYTE; Current number of PSKs allocated	: D$PCR
var	*$PCRPSKM*	BYTE; Maximum number of PSKs allowed	: D$PCR
var	*$PCRSECF*	[0] BYTE; First byte of PCR sector	: D$PCR
var	*$PCRSTAT*	[38] BYTE; State storage area	: D$PCR
var	*$PCRSTTE*	UNSIGNED; State flags	: D$PCR
var	*$PCRTOP*	[0] BYTE; First byte after top of PCR	: D$PCR
var	*$PCRYES*	CHAR; DEFINE($YES, 'Y')	: D$PCR
abr	*PDAM*	Physical Device Access Module	
abr	*PDC*	Packed Decimal Code	
fld	*$PDONE*	BYTE; Number of sectors done	: $PFDB
abr	*PFDB*	Physical File Descriptor Block	
typ	*$PFDB*	STRUCT; Physical File Descriptor Block	: D$RMS

typ	$PFDBBUF	STRUCT; PFDB Buffer List Entry	: D$RMS	
fld	$PFVID	UNSIGNED; File access variable ID	: $FILESTBL	
fld	$PFVID	UNSIGNED; 16-bit FAV Identification	: $PFDB	
fld	$PFVID	UNSIGNED; 16-bit FAV identifier	: $SECURETBL	
abr	PIO	Parallel I/O		
var	PIPEDIO$	EXTERN PIPEDIO$ BYTE; Driver function	: D$UFRWFIO	
	PIPEDIO$	WFIO to Pipe Double Buffered instead of to disk		
sc	$PIPEGEN	Create a Pipe Resource	: D$RMSIO	
typ	$PIPEGENPT	STRUCT; Pipe Generation Parameter Table	: D$RMSIO	
fld	$PIPEIAC	$ACCODES; Initial access code	: $PIPEGENPT	
var	PIPEIO$	EXTERN PIPEIO$ BYTE; Driver function	: D$UFRWFIO	
	PIPEIO$	WFIO to Pipe instead of to disk		
fld	$PIPEKEY	$FILEKEYS; Keys	: $PIPEGENPT	
fld	$PIPENAM	$NAMET; Name	: $PIPEGENPT	
fld	$PIPETERM	BYTE; Key list terminator	: $PIPEGENPT	
sc	$PIPEUSE	Check Local Pipe-in-use Status	: D$RMSIO	
var	PIPTIMO$	EXTERN PIPTIMO$ BYTE; Driver ?	: D$UFRWFIO	
	PIPTIMO$	Pipe Timeout Byte (Not Driver?)		
fld	$PMXBF	BYTE; Maximum number of buffers	: $PFDB	
abr	PNTR	Pointer		
abr	POS	Position		
abr	PRI	Primary		
val	$PRIMAX	007 "LOWEST" priority level	: D$RMSPROG	
val	$PRINORM	004 "NORMAL" priority level ($NRPRIOR/2)	:D$RMSPROG	|*
abr	PRIVP	Private Response Pipe		
abr	PRN	Physical Record Number		
abr	PRNTR	Printer		
var	PRTDIO$	EXTERN PRTDIO$ BYTE; Driver function	: D$UFRWFIO	
	PRTDIO$	WFIO to Printer Double Buffered instead of to disk		
var	PRTIO$	EXTERN PRTIO$ BYTE; Driver function	: D$UFRWFIO	
	PRTIO$	WFIO to Printer instead of to disk		
fld	PSK	BYTE; PSK or -1	: $PFDBBUF	
abr	PSK	Physical Sector Key; Points to 4K Byte mem sector		
fld	$PSUBF	BYTE; Sub-functon code: TAPE	: $PFDB	
val	$PSUFCNV	0200 Convert mode flag, Write (pack) or		
	$PSUFCNV	Read (unpack) 7-track tape data	: D$RMSIO	
val	$PSUFMSK	0177 $PSUBF function mask	: D$RMSIO	
fld	$PSYSCODE	BYTE; DIRECT RIM system code	: $PFDB	
fld	$PTASKID	BYTE; DIRECT RIM task ID	: $PFDB	
val	$PTFBSPB	003 Backspace to previous block	: D$RMSIO	
val	$PTFBSPF	005 Backspace to previous TAPE-MARK	: D$RMSIO	
val	$PTFERAS	002 Erase some (3.5 inches) tape	: D$RMSIO	
	$PTFERAS	Write extended inter-record gap	: D$RMSIO	|*
val	$PTFFSPB	002 Forward space (skip)to next block	: D$RMSIO	
val	$PTFFSPF	004 Forward space to next tape-mark	: D$RMSIO	
val	$PTFRDRV	001 Block read reverse	: D$RMSIO	
val	$PTFREAD	000 Block read	: D$RMSIO	
val	$PTFRWND	006 Rewind the tape, GO READY	: D$RMSIO	
val	$PTFULOD	007 Rewind and unload the tape	: D$RMSIO	
val	$PTFWRIT	000 Block write	: D$RMSIO	
val	$PTFWRTM	001 Write a tape-mark	: D$RMSIO	
fld	$PTIMER	BYTE; Timeout count:PIPE,DIRECT RIM	: $PFDB	
fld	$PTODO	BYTE; Number of sectors to do	: $PFDB	
abr	PTR	Pointer		
val	$PTRFLDV	66 DEFINE'd Printer Form Length Default Value	:D$PCR	

ufr	*$PUTELGX*	Log Message	: D$UFR
ufr	*$PUTELOG*	Home-Down Roll, Log Error Message	: D$UFR
ufr	*$PUTERP*	Home-Down Roll, Log/Display $MSG, Error	: D$UFR
ufr	*$PUTERPX*	Log/Display from $MSG, Error	: D$UFR
ufr	*$PUTERRR*	Home-Down Roll, Log/Display, Error	: D$UFR
ufr	*$PUTERRX*	Log/Display Message, Error	: D$UFR
ufr	*$PUTLINE*	Home-Down Roll, Display Message	: D$UFR
ufr	*$PUTLINX*	Display Message	: D$UFR
ufr	*$PUTLNP*	Home-Down Roll, Display from $MSG	: D$UFR
ufr	*$PUTLNPX*	Display from $MSG	: D$UFR
ufr	*$PUTLOG*	Home-Down Roll, Log Message	: D$UFR
ufr	*$PUTLOGX*	Log Message	: D$UFR
ufr	*$PUTNOP*	Home-Down Roll, Log/Display from $MSG	: D$UFR
ufr	*$PUTNOPX*	Log/Display from $MSG	: D$UFR
ufr	*$PUTNOTE*	Home-Down Roll, Log/Display Message	: D$UFR
ufr	*$PUTNOTX*	Log/Display Message	: D$UFR
var	*PUTWSMD$*	BYTE EXTERN; $PUTx UFRs, WSIO mode($WSM..)	: D$UFR
abr	*PWS*	?	
abr	*RAM*	Random Access Memory	
def	*RASLEND$*	Turn off RASL Traps in program running RASL	:D$I
def	*RASLRES$*	Invoke the DASL Debugger	: D$I
val	*$RCEPN*	0206 Entry pnt mem in Rel Lib:$RELCODE	:D$RMSSTRU
val	*$RCEXDEF*	0203 External Definitions : $RELCODE	: D$RMSSTRU
val	*$RCEXREF*	0204 External References : $RELCODE	: D$RMSSTRU
abr	*RCL*	Remote Connection Link	
val	*$RCLINEN*	0207 DEBUG Information : $RELCODE	: D$RMSSTRU
val	*$RCOBJ*	0202 Object Text : $RELCODE	: D$RMSSTRU
val	*$RCPID*	0201 Program Identification:$RELCODE	: D$RMSSTRU
val	*$RCXFER*	0205 Trnasfer Address :$RELCODE	: D$RMSSTRU
val	*$RD*	0203 Roll screen down one line	: D$RMS
ddw	*RECURSIVE*	Specifies Function may be Called Recursively	
abr	*REL*	Relocatable	
typ	*$RELCODE*	BYTE; Relocatable Sector Type Codes	: D$RMSSTRU
typ	*$RELEPN*	UNION; Rel Entry Point Member Entry	: D$RMSSTRU
fld	*$RELEPNAME*	[8] CHAR; Normal entry point name	: $RELE
fld	*$RELEPNMBR*	STRUCT; New program entry	: $RELE
fld	*$RELEPNMFLG*	CHAR; New program flag	: $RELE
fld	*$RELEPNMLSN*	UNSIGNED; Program LSN	: $RELE
fld	*$RELEPNMSKP*	[5] BYTE;	: $RELE
typ	*$RELEPNS*	[31] $RELEPN;Rel Entry Point Mbr Secr	: D$RMSSTRU
sc	*$RELFAVS*	?	: D$RMSSI
fld	*$RELFWDFLAG*	BYTE; forward reference flag	: $RELXRE
fld	*$RELFWDOFFS*	UNSIGNED; Offset	: $RELXRE
fld	*$RELFWDPAB*	BYTE; PAB	: $RELXRE
fld	*$RELFWDSKIP*	[4] BYTE;	: $RELXRE
typ	*$RELLINE*	[84] $RELLNENT;Rel DEBUG Line Nbrs Sec	:D$RMSSTRU
typ	*$RELLNENT*	Rel DEBUG Line Number Entry	: D$RMSSTRU
typ	*$RELOBJ*	[255] BYTE; Rel Object Code Sector	: D$RMSSTRU
typ	*$RELPID*	Rel Program ID Sector	: D$RMSSTRU
fld	*$RELSECTOR*	Rel sectors	: $LIBSECT
typ	*$RELXDEF*	[22] $RELXDENT;Rel Extern Def Sector	: D$RMSSTRU
typ	*$RELXDENT*	Rel External Definition Entry	: D$RMSSTRU
typ	*$RELXFER*	Rel Starting Address Sector	: D$RMSSTRU

fld	***$RELXFEROFFS***	UNSIGNED; Offset	: $RELXFER
fld	***$RELXFERPAB***	BYTE; PAB	: $RELXFER
typ	***$RELXREF***	[31] $RELXRENT; Rel Extern Ref Entry	: D$RMSSTRUCT
typ	***$RELXRENT***	UNION; Rel External Reference Entry	: D$RMSSTRUCT
fld	***$RELXREXNAM***	[8] CHAR; name	: $RELXRENT
fld	***$RELXRFWD***	STRUCT; Forward reference definition	: $RELXRENT
sc	***$RENENV***	Change a Disk File Name	: D$RMSIO
sc	***$REOPEN***	Reopen a File With New Passwords	: D$RMSIO
abr	***REQ***	Request	
dcw	***RESULT***	Assign a value to the RESULT of a function.	
	$RFI	DASL Macro for defining Macros	: D$RMSPROG
sc	***$RFIAKS***	Return From Abort Key Seq Interrupt	: D$RMSPROG
sc	***$RFIDKS***	Return from $TRAPDKS Interrupt	: D$RMSPROG
sc	***$RFIFK***	Return from $TRAPFK Interrupt	: D$RMSPROG
sc	***$RFIKKS***	Return From $TRAPKKS Interupt	: D$RMSPROG
sc	***$RFILKS***	Return from LOG-OFF Trap Seq Interupt	: D$RMSPROG
val	***$RFITRC***	0200 $RFI trap remain clear bit	: D$RMSPROG
abr	***RI***	Receiver Inhibited	
abr	***RIM***	Resource Interface Module	
fld	***$RLADDR***	^ BYTE; Starting load address	: $RLPARAM
typ	***$RLDEF***	STRUCT; Rel Loader Definition Structure	: D$UFRRLD
fld	***$RLDEFAD***	^ $RLDEF; User definition table address	: $RLPARAM
fld	***$RLDLIM***	^ $RLDEF; User def table limit address	: $RLPARAM
fld	***$RLDNAME***	$RLNAME; Name	: $RLDEF
fld	***$RLDVAL***	UNSIGNED; Value	: $RLDEF
val	***$RLFEXDO***	004 External definition full : $RLFLAGS	: D$UFRRLD
fld	***$RLFLAG***	$RLFLAGS; Load control flag	: $RLPARAM
typ	***$RLFLAGS***	Relocatable Loader Flags	: D$UFRRLD
val	***$RLFNSVD***	001 Supress Definition storage : $RLFLAGS	: D$UFRRLD
val	***$RLFRFUL***	010 Reference work area full : $RLFLAGS	: D$UFRRLD
val	***$RLFTOPD***	002 Top-down memory load : $RLFLAGS	: D$UFRRLD
val	***$RLFUNDF***	020 Undefined symbol encounterd: $RLFLAGS	: D$UFRRLD
fld	***$RLLIM***	^ BYTE; Limiting load address	: $RLPARAM
fld	***$RLLSN***	UNSIGNED; Member logical sector number	: $RLPARAM
typ	***$RLNAME***	[8] CHAR; Relocatable Loader Name Type	: D$UFRRLD
fld	***$RLNEOFFS***	UNSIGNED; Line number offset	: $RELLNENT
fld	***$RLNEPAB***	BYTE; Line number PAB	: $RELLNENT
typ	***$RLPARAM***	STRUCT; Relocatable Loader Parameters	: D$UFRRLD
fld	***$RLPFDB***	^ $PFDB; Phys file descriptor block adr	: $RLPARAM
typ	***$RLREF***	UNSIGNED; Rel Loader Reference Work Area	: D$UFRRLD
fld	***$RLREFAD***	^ $RLREF; User reference work area adr	: $RLPARAM
fld	***$RLRLIM***	^ $RLREF; Limit adr of ref work area	: $RLPARAM
fld	***$RLUDRTN***	^ $RLUDRTNF; User defined symbol routine	: $RLPARAM
typ	***$RLUDRTNF***	(); Relocatable Loader User Routine Type	: D$UFRRLD
fld	***$RLUMEMA***	^ $RLUMEMAF; User memory allocatn routine	: $RLPARAM
typ	***$RLUMEMAF***	(); Relocatable Loader User Routine Types	: D$UFRRLD
abr	***RMS***	Resource Management System	
ufr	***$RMSMSG***	Err-Msg, Return RMS Message	: D$UFRERR
abr	***ROM***	Read Only Memory	
fld	***$RPDPABTABLE***	[16] $PABENTRY; PAB table	: $RELPID
fld	***$RPDPGMNAM***	$LNAMET; Program name	: $RELPID
fld	***$RPDXDPTR***	UNSIGNED; External definition pointer	: $RELPID
abr	***RSP***	in NQDQ ?	
	RSP	context functions: 16 error during $SECW to RSP ?	
typ	***$RSRCFLAGS***	$INFO Resource Flags	: D$RMSGEN

val	*$RU*	0202 Roll screen up one line	: D$RMSWS
ufr	*$$RUN*	Interface to $RUN System Call	: D$UFRWS
sc	*$RUN*	Load and Run a Program	: D$RMSPROG
abr	*RX*	Receive	
fld	*$RXDENAME*	[8] CHAR; name	: $RELXDENT
fld	*$RXDEPAB*	BYTE; PAB	: $RELXDENT
fld	*$RXDEVAL*	UNSIGNED; value	: $RELXDENT

abr	*SC*		System Call	
val	*SC$BASESET*	12	System Call ErrNum	: D$ERRNUM
val	*SC$CLOSE*	25	System Call ErrNum	: D$ERRNUM
val	*SC$CLOSEAL*	47	System Call ErrNum	: D$ERRNUM
val	*SC$DISCONT*	60	System Call ErrNum	: D$ERRNUM
val	*SC$DONATFV*	63	System Call ErrNum	: D$ERRNUM
val	*SC$ERROR*	43	System Call ErrNum	: D$ERRNUM
val	*SC$EXIT*	42	System Call ErrNum	: D$ERRNUM
val	*SC$FILES*	29	System Call ErrNum	: D$ERRNUM
val	*SC$FORMAT*	23	System Call ErrNum	: D$ERRNUM
val	*SC$GETIME*	32	System Call ErrNum	: D$ERRNUM
val	*SC$GLUTEN*	34	System Call ErrNum	: D$ERRNUM
val	*SC$INFO*	35	System Call ErrNum	: D$ERRNUM
val	*SC$LOAD*	45	System Call ErrNum	: D$ERRNUM
val	*SC$LOCKFAV*	64	System Call ErrNum	: D$ERRNUM
val	*SC$MEMCTL*	15	System Call ErrNum	: D$ERRNUM
val	*SC$MEMGET*	8	System Call ErrNum	: D$ERRNUM
val	*SC$MEMKEY*	11	System Call ErrNum	: D$ERRNUM
val	*SC$MEMMAP*	10	System Call ErrNum	: D$ERRNUM
val	*SC$MEMPROT*	13	System Call ErrNum	: D$ERRNUM
val	*SC$MEMREL*	9	System Call ErrNum	: D$ERRNUM
val	*SC$OPENENV*	24	System Call ErrNum	: D$ERRNUM
val	*SC$PIPEGEN*	36	System Call ErrNum	: D$ERRNUM
val	*SC$PIPEUSE*	61	System Call ErrNum	: D$ERRNUM
val	*SC$RELFAVS*	65	System Call ErrNum	: D$ERRNUM
val	*SC$RENENV*	28	System Call ErrNum	: D$ERRNUM
val	*SC$REOPEN*	30	System Call ErrNum	: D$ERRNUM
val	*SC$RFI*	41	System Call ErrNum	: D$ERRNUM
val	*SC$RUN*	44	System Call ErrNum	: D$ERRNUM
val	*SC$SECCHK*	21	System Call ErrNum	: D$ERRNUM
val	*SC$SECEOF*	26	System Call ErrNum	: D$ERRNUM
val	*SC$SECR*	16	System Call ErrNum	: D$ERRNUM
val	*SC$SECRO*	17	System Call ErrNum	: D$ERRNUM
val	*SC$SECURE*	27	System Call ErrNum	: D$ERRNUM
val	*SC$SECW*	18	System Call ErrNum	: D$ERRNUM
val	*SC$SECWAIT*	20	System Call ErrNum	: D$ERRNUM
val	*SC$SECWO*	19	System Call ErrNum	: D$ERRNUM
val	*SC$SETIME*	33	System Call ErrNum	: D$ERRNUM
val	*SC$SETMAX*	75	System Call ErrNum	: D$ERRNUM
val	*SC$SETMIN*	14	System Call ErrNum	: D$ERRNUM
val	*SC$SETPRI*	46	System Call ErrNum	: D$ERRNUM
val	*SC$SETSQL*	38	System Call ErrNum	: D$ERRNUM
val	*SC$SIGNON*	73	System Call ErrNum	: D$ERRNUM
val	*SC$SINDEP*	48	System Call ErrNum	: D$ERRNUM
val	*SC$SLOCAL*	49	System Call ErrNum	: D$ERRNUM
val	*SC$STOPIO*	37	System Call ErrNum	: D$ERRNUM

val	*SC$TASKCTL*	55 System Call ErrNum	: D$ERRNUM	
val	*SC$TIMER*	39 System Call ErrNum	: D$ERRNUM	
val	*SC$TRAPSET*	40 System Call ErrNum	: D$ERRNUM	
val	*SC$UCSCHK*	53 System Call ErrNum	: D$ERRNUM	
val	*SC$UCSDEL*	54 System Call ErrNum	: D$ERRNUM	
val	*SC$UCSGEN*	50 System Call ErrNum	: D$ERRNUM	
val	*SC$UCSSIG*	51 System Call ErrNum	: D$ERRNUM	
val	*SC$UCSWAIT*	52 System Call ErrNum	: D$ERRNUM	
val	*SC$USRABN*	74 System Call ErrNum	: D$ERRNUM	
val	*SC$WAITIO*	22 System Call ErrNum	: D$ERRNUM	
val	*SC$WAITIOS*	66 System Call ErrNum	: D$ERRNUM	
val	*SC$WCONFIG*	0 System Call ErrNum	: D$ERRNUM	
val	*SC$WSCTL*	4 System Call ErrNum	: D$ERRNUM	
val	*SC$WSGETCH*	2 System Call ErrNum	: D$ERRNUM	
val	*SC$WSIO*	5 System Call ErrNum	: D$ERRNUM	
val	*SC$WSTATUS*	1 System Call ErrNum	: D$ERRNUM	
val	*SC$WSWAIT*	3 System Call ErrNum	: D$ERRNUM	
ufr	*$SCANCFG*	Read and Scan User Conf File	: D$UFRSCAN	
ufr	*$SCANFLS*	Scan File Specs According to Table	: D$RMS	
ufr	*$SCANFS*	Scan a File Specification	: D$UFRSCAN	
ufr	*$SCANHSI*	Compress HSI	: D$UFRENV	
ufr	*$SCANNB*	Scan to Next Non-Blank	: D$UFRSCAN	
ufr	*$SCANOS*	Scan Options Specification	: D$RMS	
var	*SCANPTR$*	EXTERN SCANPTR$ ∧ CHAR; ?	: D$UFRSCAN	
ufr	*$SCANSYM*	CmdInt, Scan a Symbol	: D$UFRSCAN	
ufr	*$SCANWRD*	CmdInt, Scan Two Word Lists for Matches	: D$UFRSCAN	
abr	*SCF*	System Configuration File		
typ	*$SCFMSCODES*	ENUM(...); FAR Exception Exit Codes	: D$FAR	
abr	*SDF*	Subrecord Definitions File		
ufr	*$SDIVID3*	Numeric, Signed 24-bit Division	: D$UFRNUM	
abr	*SEC*	Sector		
val	*$SECACT*	002 Operation still in progress:$SECSTAT	: D$RMSIO	|*
val	*$SECBSD*	0200 Resource is a byte strg dev:$SECSTAT	:D$RMSIO	
sc	*$SECCHK*	Check Operation Status	: D$RMSIO	
sc	*$SECEOF*	Obtain or Set End Of File Location	: D$RMSIO	
val	*$SECERR*	004 Soft recoverable error : $SECSTAT	: D$RMSIO	|*
val	*$SECFLOK*	020 FAV locked across CLOSEALLs ($LOCKFAV)		
val	*$SECFLOK*	: $SECSTAT	: D$RMSIO	
fld	*SECONDS*	$TIME; Seconds since beginning of 1901	: $SYSTIME	
sc	*$SECR*	Block Read	: D$RMSIO	
sc	*$SECRO*	Block Read Optimum	: D$RMSIO	
val	*$SECSS*	0100 Stop I/O has been sent : $SECSTAT	: D$RMSIO	
typ	*$SECSTAT*	Block I/O Status Control	: D$RMSIO	
val	*$SECSTOP*	040 I/O stopped by user : $SECSTAT	: D$RMSIO	
val	*$SECTMD*	010 Tape-mark detected : $SECSTAT	: D$RMSIO	
sc	*$SECURE*	Multi-Resource, Change Disk File Security	: D$RMSIO	
typ	*$SECURETBL*	STRUCT; $SECURE Parameter Table	: D$RMSIO	
sc	*$SECW*	Block Write	: D$RMSIO	
sc	*$SECWAIT*	Wait for Operation Complete	: D$RMSIO	
sc	*$SECWO*	Block Write Optimum	: D$RMSIO	
val	*$SECWP*	1 Resource physical write protect:$SECSTAT	:D$RMSIO	
dim	*SET*	Define Var Type BYTE and Define Powers of 2	:D$INC	
ufr	*$SETABTF*	Set CHAIN Abort Flag	: D$UFRWS	
fld	*$SETADJ*	[3] BYTE; Clock adj in seconds per day	: $SETTIMEP	
fld	*$SETDIR*	BYTE; Clock adjustment direction	: $SETTIMEP	

sc	***$SETIME***	Set The Current System Time	: D$RMSGEN
sc	***$SETMAX***	Set maximum memory requirement	: D$RMSMEM
sc	***$SETMIN***	Set Minimum Memory Requirement	: D$RMSMEM
sc	***$SETPRI***	Set User Task Priority Level	: D$RMSPROG
fld	***$SETSEC***	$TIME;Time in secnds since begin of 1901	:$SETTIMEP
sc	***$SETSQL***	Set Security Level	: D$RMSGEN
typ	***$SETTIMEP***	STRUCT; $SETIME Parameter Table	: D$RMSGEN
dim	***SETV***	Define Ascending Powers of 2 from Initial	: D$INC
dim	***SETW***	Define Var Type UNSIGNED; Powers of 2	: D$INC
typ	***$SFENT***	STRUCT; Symbolic File Table	: D$RMS
fld	***$SFIP***	∧ $SFITABLE;	: $FILESTBL
typ	***$SFITABLE***	STRUCT; $GETSFI Information	: D$RMSIO
typ	***$SFNT***	[8] CHAR; Symbolic Field Name	: D$RMS
fld	***$SFTENV***	$ENVN; File environment	: $SFENT
fld	***$SFTEXT***	$EXTT; File extension	: $SFENT
fld	***$SFTNAM***	$NAMET; File name	: $SFENT
fld	***$SFTSFN***	$SFNT; Symbolic field name	: $SFENT
abr	***SID***	Source Identification	
sc	***$SIGNON***	Force User Signon	: D$RMSPROG
sc	***$SINDEP***	Start an Independent Task	: D$RMSTASK
drw	***SIZEOF***	Operator Which Gives Size in Bytes of Argument	
val	***$SKAFX***	1 Peripheral FAX device	: D$RMSIO
val	***$SKCM38I***	006 Internal com adaptor in 1800/3800	: D$RMSIO
val	***$SKCM400***	002 Async com adaptors: 9400, 9401, 9402	: D$RMSIO
val	***$SKCM401***	003 Async com 9400 with Bell 103 modem	: D$RMSIO
val	***$SKCM402***	004 Async com 9400 with Bell 202 modem	: D$RMSIO
val	***$SKCM462***	005 Port on multi port: 9462 MPCA port	: D$RMSIO
val	***$SKCM481***	001 Multi function com adaptor: 9481 MFCA	: D$RMSIO
val	***$SKCM86I***	007 Internal com adaptor in 8600	: D$RMSIO
val	***$SKCM86M***	9 Comm Device on 8600 MPCA	: D$RMSIO
val	***$SKCM88I***	010 Internal com adaptor in 8800	: D$RMSIO
val	***$SKDK134***	006 9390 Disk: 134 MB fixed on 8800 IMOD	: D$RMSIO
val	***$SKDK67***	005 9390: 67MB on 55/66MIDS or 8800IMOD	: D$RMSIO
val	***$SKDKC10***	010 9310: 10 MB Cynthia on 8600 Microbus	: D$RMSIO
val	***$SKDKCTG***	002 9350: 2.5 MB Cartridge on 5500/6600	: D$RMSIO
val	***$SKDKF01***	009 1403: 1 MB Dbl-Side Dbl-Dens Diskette	: D$RMSIO
	$SKDKF01	"TORTILLA",on 8600 uBus	
val	***$SKDKFLX***	001 Flexible diskette on 1800 Microbus	: D$RMSIO
val	***$SKDKM10***	003 9374: 10 MB Mass storage on 5500/6600	: D$RMSIO
val	***$SKDKM20***	007 9301:20MB fxd disk on 8600PIO "Whizzie"	:D$RMSIO
val	***$SKDKM25***	004 9370: 25 MB Mass storage on 5500/6600	: D$RMSIO
val	***$SKDKS10***	11 10 MB Moses on 8600 micro-buss	: D$RMSIO
val	***$SKDKS40***	12 40 MB Moses on 8600 micro-buss	: D$RMSIO
val	***$SKDKW10***	10 9315: 10 MB Cyclone on 8600 micro-Buss	:D$RMSIO
val	***$SKDKW20***	015 20 MB 'Cyclone' on 8600 micro-Buss	: D$RMSIO |
val	***$SKFCI***	0 FAX communications interface	: D$RMSIO
val	***$SKMT75***	004 7-track 556 BPI Tape	: D$RMSIO
val	***$SKMT78***	001 7-track 800 BPI Tape	: D$RMSIO
val	***$SKMT916***	003 9-track 1600 BPI Tape	: D$RMSIO
val	***$SKMT98***	002 9-track 800 BPI Tape	: D$RMSIO
val	***$SKPT297***	011 132 Col Serial printer attached to MP	: D$RMSIO
val	***$SKPT601***	006 Mercury serial printer attachd to MPCA	:D$RMSIO
val	***$SKPT611***	007 Orion serial printer attached to MPCA	: D$RMSIO
val	***$SKPT621***	010 Freedom serial printer attachd to MPCA	:D$RMSIO
val	***$SKPTFRE***	002 Freedom printer	: D$RMSIO

val	*$SKPTFST*	003 Freedom prntr with secondary tractor	: D$RMSIO
val	*$SKPTLOC*	001 Line printer	: D$RMSIO
val	*$SKPTMER*	005 Mercury printer parallel I/O	: D$RMSIO
val	*$SKPTSVO*	004 Servo printer	: D$RMSIO
val	*$SKUNDEF*	000 Resource has no subkind	: D$RMSIO
val	*$SKWS36*	003 3601/8200 ver 1.1 Multiport Terminal	: D$RMSIO
val	*$SKWS38*	002 1800/3800 processor console	: D$RMSIO
val	*$SKWS56*	001 5500/6600 processor console	: D$RMSIO
val	*$SKWS822*	006 8220 workstation	: D$RMSIO
val	*$SKWS823*	007 8230 Workstation	: D$RMSIO |*
val	*$SKWS86*	005 8600 processor console	: D$RMSIO
val	*$SKWSALN*	010 Alien device (non Datapoint)	: D$RMSIO
val	*$SKWSNA*	000 Workstation not available	: D$RMSIO
val	*$SKWSRMS*	04 RMS WS (8200 ver.2 multi-port terminal)	:D$RMSIO
sc	*$SLOCAL*	Start Local Task	: D$RMSTASK
ufr	*$SMLPLY3*	Numeric, Signed 24-bit Multiplication	: D$UFRNUM
abr	*SOH*	Start of Header	
typ	*$SONT*	[8] CHAR; Symbolic Option Field Name	: D$RMS
abr	*SPRM*	RMS System Programmer's Reference Manual (Vols 1-4)	
abr	*SPV*	?	
abr	*SPV*	Shared Program Variable	
abr	*SQL*	Security Level	
val	*$SQLCHEK*	010 Security level required to check	: D$RMSGEN
val	*$SQLMAX*	011 Highest possible security level	: D$RMSGEN
val	*$SQLREPR*	011 $SQLMAX:Security lvl requird to repair	:D$RMSGEN
fld	*$SSALLO*	$LSN; LSN of last allocated sector	: $SECURETBL
fld	*$SSFCT*	$TIME; File creation time	: $SECURETBL
fld	*$SSFFMT*	BYTE; File format code	: $SECURETBL
fld	*$SSFIAC*	$ACCODES; File initial access code	: $SECURETBL
fld	*$SSFINC*	UNSIGNED; File increment	: $SECURETBL
fld	*$SSFSL*	BYTE; File security level	: $SECURETBL
val	*$SSGET*	000 File security get from FDT	: D$RMSIO
val	*$SSGETX*	002 Get extra info from FDT	: D$RMSIO
fld	*$SSKEYS*	$FILEKEYS; access keys and codes:$SECURETBL	:D$RMSIO
val	*$SSPUT*	001 File security put into FDT	: D$RMSIO
val	*$SSPUTX*	003 Put extra info into FDT	: D$RMSIO
fld	*$SSSEGM*	BYTE; Number of segments in file	: $SECURETBL
fld	*$SSX*	[2] BYTE;	: $SECURETBL
typ	*$STARTADR*	^ D$CALLF; Starting Address Type	: D$RMS
ddw	*STATIC*	Prevents Re-allocation of Var. in Recur. Function	
val	*$STOPALL*	000001 Stop all I/O	: D$RMSIO
sc	*$STOPIO*	Stop All Data Movement	: D$RMSIO
val	*$STOPONE*	000000 Stop I/O given by PFDB	: D$RMSIO
ddw	*STRUCT*	Named Member Consisting of Several Named Members	
dcm	*SUBSTR*	Select Part of String, Begin at Start for Length	
fld	*SYS*	UNSIGNED; System usage	: $PFDBBUF
drw	*SYSTEM*	Reserved for Future Code Generators	
typ	*$SYSTIME*	STRUCT; System Time	: D$RMSGEN
typ	*$SYSTINFO*	System Time Information	: D$RMSGEN

abr	*TA*	Transmitter Available		
var	*TAPEDIO$*	EXTERN TAPEDIO$ BYTE; Driver	D$UFRWFIO	
	TAPEDIO$	Tape Double Buffered I/O Driver Routine		
var	*TAPEIO$*	EXTERN TAPEIO$ BYTE; Driver	D$UFRWFIO	
	TAPEIO$	Tape Single Buffered I/O Driver Routine		
ufr	*$TAPEREWIND*	Rewind Tape	D$UFRWFIO	
ufr	*$TAPEUNLOAD*	Rewind Tape and Unload	D$UFRWFIO	
sc	*$TASKCTL*	Exert Control Over a Task	D$RMSTASK	
abr	*TBD*	Broadcast Delay Time		
val	*$TCAKS*	1 Force abort	D$RMSTASK	
abr	*TCB*	Task Control Block		
val	*$TCDKS*	2 Force DISPLAY key sequence trap	D$RMSTASK	
val	*$TCFK*	4 Force FUNCTION key trap	D$RMSTASK	
val	*$TCKKS*	3 Force KEYBOARD key sequence trap	D$RMSTASK	
val	*$TCLOKS*	0 Force LOGOFF	D$RMSTASK	
val	*$TCVAKS*	6 Force VANTAGE ABORT Key Seq Trap	D$RMSTASK	
val	*$TCVLKS*	5 Force VANTAGE LOG-OFF Key Seq Trap	D$RMSTASK	
dcw	*THEN*	Part of IF THEN ELSE Execution Control		
typ	*$TIME*	[5] BYTE; Time in Secnds Since Begin of 1901	D$RMS	
sc	*$TIMER*	Reset System Timer	D$RMSPROG	
abr	*TLPM*	Tape Label Processing Module		
fld	*$TM8MS*	BYTE; Eight Millisecond Counter	$SYSTINFO	
abr	*TMA*	Transmitted Message Acknowledged		
fld	*$TMADJ*	[3] BYTE; Clock Adj'ment in Seconds/Day	$SYSTINFO	|*
val	*$TMADJDN*	0 Adjust Clock Down : $SYSTINFO	D$RMSGEN	|*
fld	*$TMADJDR*	BYTE; Clock Adjustment Direction	$SYSTINFO	
val	*$TMADJUP*	2 Adjust Clock Up : $SYSTINFO	D$RMSGEN	
fld	*$TMDSTDC*	BYTE; Number of Days to Count	$DSTINFO	
val	*$TMDSTDI*	1 Direction Flag: : $DSTINFO	D$RMSGEN	|*
	$TMDSTDI	0=From Start 1=From End		
fld	*$TMDSTFG*	DST Start/Stop Flags	$DSTINFO	
fld	*$TMDSTHR*	BYTE; Hour Start/Stop DST	$DSTINFO	
fld	*$TMDSTMN*	BYTE; Month to Start/End DST	$DSTINFO	
fld	*$TMDSTOS*	BYTE; DST Adjustment in Hours	$SYSTINFO	
fld	*$TMDSTWD*	BYTE; Day of Week to Start/Stop DST	$DSTINFO	
fld	*$TMDTEND*	$DSTINFO; DST End Table	$SYSTINFO	
fld	*$TMDTSTR*	$DSTINFO; DST Start Table	$SYSTINFO	
fld	*$TMTZ*	$TIME; Time Zone Offset from UTC	$SYSTINFO	
fld	*$TMUTC*	$TIME; Universal Co-ordinated Time	$SYSTINFO	
abr	*TPM*	Message Propagation Time		
abr	*TPT*	Token Propagation Time		
sc	*$TRAPAKS*	Trap ABORT Key Sequence	D$RMSPROG	
sc	*$TRAPDKS*	Trap DISPLAY-CANCEL-DISPLAY Key Sequence	D$RMSPROG	
sc	*$TRAPFK*	Trap Function Key Strokes	D$RMSPROG	
sc	*$TRAPKKS*	Trap Keyboard Key Sequence	D$RMSPROG	
sc	*$TRAPLKS*	Trap LOG-OFF Key Sequence	D$RMSPROG	
sc	*$TRAPSET*	Trap Set System Call, Used Indirectly	D$RMSPROG	
sc	*$TRAPUMV*	Trap User Mode Violations.	D$RMSPROG	
abr	*TRC*	Recovery Time		
abr	*TRP*	Response Timeout		
val	*TRUE*	ENUMV 001 Boolean value: 'true'	D$INC	
abr	*TSB*	Third Significant Byte (bits 16-23)		
abr	*TSF*	Task Status File		
abr	*TTA*	Turnaround Time		
abr	*TX*	Transmit		

ufr	***$TXBKSP***	Backspace a logical record : D$UFRWFIO
ufr	***$TXCLOSE***	Terminate Processing for a Text-File : D$UFRWFIO
ufr	***$TXDEL***	Delete a Logical Text Record : D$UFRWFIO
ufr	***$TXOPEN***	Prepare an Opened Text-File for Access : D$UFRWFIO
ufr	***$TXOPENP***	Open Using Specified Physical I/O Routine:D$UFRWFIO
ufr	***$TXPOSEF***	Position to Text-File EOF : D$UFRWFIO
ufr	***$TXPOSIT***	Position Text-File to File Pointer : D$UFRWFIO
ufr	***$TXPREP***	Prepare a New Text-File for Access : D$UFRWFIO
ufr	***$TXPREPP***	Prepare Using Specified Phys I/O Routine :D$UFRWFIO
ufr	***$TXREAD***	Read a Logical Text-File Record : D$UFRWFIO
ufr	***$TXUPDATE***	Update a Logical Text Record : D$UFRWFIO
ufr	***$TXWEOF***	Write EOF at Current Text-File Position : D$UFRWFIO
ufr	***$TXWRITB***	Write Text-file record, specified length :D$UFRWFIO |*
ufr	***$TXWRITE***	Write a Logical Text-File Record : D$UFRWFIO
ddw	***TYPDEF***	Give a Type a Name that may be Used as a Type
abr	***U-BUSS***	Micro-Buss
fld	***$UABAKSS***	UNSIGNED; ABORT key seq trap addr : $UABSECTOR
val	***$UABALOC***	1 Mem allocation table LSN :$UABSECTOR: D$RMSSTRUCT
fld	***$UABBR***	BYTE; User base register : $UABSECTOR
val	***$UABCLR***	2 De-activate User ABEND for this task : D$RMSPROG
fld	***$UABDAD***	$NAMET; Father task name : $UABSECTOR
fld	***$UABDATA***	[256] BYTE; LSN 3 thru n : $UABSECTOR
fld	***$UABDKSS***	UNSIGNED; Display key seq trap adddr : $UABSECTOR
val	***$UABDUAL***	1 Dual Sector Tables active :$UABSECTOR:D$RMSSTRUCT
val	***$UABDUMP***	3 Memory dump LSN : $UABSECTOR : D$RMSSTRUCT
fld	***$UABERR***	BYTE; Error control : $UABSECTOR
fld	***$UABFAVA***	BYTE; Nbr of active I/O operations : $UABSECTOR
fld	***$UABFAVC***	BYTE; Nbr of complete I/O operations : $UABSECTOR
fld	***$UABFKS***	UNSIGNED; Function key trap addr : $UABSECTOR
fld	***$UABFLG***	BYTE; User flags : $UABSECTOR
val	***$UABHDR***	0 header sector LSN : $UABSECTOR : D$RMSSTRUCT
fld	***$UABKKSS***	UNSIGNED; Keyboard key seq trap addr : $UABSECTOR
val	***$UABLCL***	020 local task : $UABSECTOR.$UABSTAT: D$RMSSTRUCT
fld	***$UABLKSS***	UNSIGNED; Logoff Key Seq Trap Addr : $UABSECTOR
fld	***$UABMCUR***	BYTE; Current memory allocation : $UABSECTOR
fld	***$UABMMAX***	BYTE; Maximum Memory Allocation : $UABSECTOR
fld	***$UABMMIN***	BYTE; Minimum memory allocation : $UABSECTOR
fld	***$UABNAME***	$NAMET; User task name : $UABSECTOR
fld	***$UABNVER***	[5] BYTE; Nucleus version : $UABSECTOR
fld	***$UABPCRK***	BYTE; PCR Sector PSK : $UABSECTOR : D$RMSSTRUCT
val	***$UABPDAD***	010 PSK ownd by father task :$UABSECTOR:D$RMSSTRUCT
val	***$UABPPCR***	020 PSK is the PCR sector : $UABSECTOR: D$RMSSTRUCT
val	***$UABPPRV***	2 PSK is private to this tsk:$UABSECTOR:D$RMSSTRUCT
val	***$UABPPWS***	32 PSK is the PWS Sector : $UABSECTOR : D$RMSSTRUCT
val	***$UABPRO***	1 PSK is read only : $UABSECTOR : D$RMSSTRUCT
fld	***$UABPROC***	BYTE; Processor type : $UABSECTOR
val	***$UABPSHR***	04 PSK is shared : $UABSECTOR : D$RMSSTRUCT
fld	***$UABPSK***	BYTE; User PSK : $UABSECTOR
fld	***$UABPSKS***	[128] STRUCT { LSN 1&2 : $UABSECTOR
val	***$UABPWSA***	4 PWS Active : $UABSECTOR : D$RMSSTRUCT
fld	***$UABPWSK***	BYTE; PWS Sector PSK : $UABSECTOR : D$RMSSTRUCT
val	***$UABPWSX***	2 Executing in PWS Sector :$UABSECTOR : D$RMSSTRUCT
fld	***$UABREG***	[8] BYTE; User registers ABCDEHLX : $UABSECTOR

typ	***$UABSECTOR***	User Abend Header Sector Format	: D$RMSSTRUCT
val	***$UABSET***	0 Activate User ABEND for this task	: D$RMSPROG
val	***$UABSETO***	1 Activate User ABEND for other task	: D$RMSPROG
val	***$UABSHAR***	010 Shared program active : $UABSECTOR	: D$RMSSTRUCT
fld	***$UABSHID***	$NAMET; Shared program name	: $UABSECTOR
fld	***$UABSPSK***	Status of user PSK	: $UABSECTOR
fld	***$UABSTAT***	User state	: $UABSECTOR
fld	***$UABSTB***	[32] BYTE; User sector table	: $UABSECTOR
fld	***$UABSTBN***	BYTE; Nbr of entries in user sector tabl	:$UABSECTOR
fld	***$UABSTK***	[32] UNSIGNED; User stack	: $UABSECTOR
fld	***$UABSTKN***	BYTE; Nbr of level in user stack	: $UABSECTOR
fld	***$UABTIME***	$TIME; Time of abort	: $UABSECTOR
fld	***$UABUMVS***	UNSIGNED; User mode violation trap addr	: $UABSECTOR
abr	***UAT***	User Access Token	
abr	***UAV***	User Access Variable	
abr	***UCP***	Unique Communication Pipe	
abr	***UCS***	User Created Semaphore	
sc	***$UCSCHK***	Check a User Created Semaphore	: D$RMSTASK
sc	***$UCSDEL***	Delete a User Created Semaphore ?	: D$RMSTASK
sc	***$UCSGEN***	Generate a User Created Semaphore	: D$RMSTASK
sc	***$UCSSIG***	Signal a User Created Semaphore	: D$RMSTASK
sc	***$UCSWAIT***	Wait on a User Created Semaphore	: D$RMSTASK
abr	***UDA***	User Data Area	

NOTE: In the ***FUNCTIONS*** section descriptions of error codes, the contents of:

$ERRC.$FUNC is usually ***SC$...*** or ***$UEC...*** and
$ERRC.$CODE is usually ***$EC...nn*** or ***$UEC...nn***

The ***$EC...nn*** or ***$UEC...nn*** ends with a decimal number which is its value.

The words with a ***:**** at the end of the line in this section, ***WORDS***, are defined in ***DASL*** in the ***D$ERRCODE*** include file. You may use those words in statements testing the value of ***$ERRC.$CODE***, otherwise you must use the decimal value.

val	***$UECCHN***	000210	CHAIN,Chaining	: D$ERRNUM
val	***$UECCHNO***	000000	Missing terminator in job file	:*
val	***$UECCHN1***	000001	End of job file reached during cmd execut	:*
val	***$UECCHN2***	000002	Invalid control record in job file	:*
val	***$UECCHN3***	000003	Job file environment not found by ENVLOC$	:*
val	***$UECCHN4***	000004	Invalid record type in job file	:*
val	***$UECCHN5***	000005	Internal error in chain execution overlay	:*
val	***$UECCHN6***	000006	Invalid header record in job file	:*
val	***$UECCHN7***	000007	CHAIN vector err when executn ovl loaded	:*
val	***$UECCHN8***	000010	Internal error in CHAIN compilation	:*
val	***$UECENVO***	000000	Invalid UET entry format	:*
val	***$UECENV1***	000001	Non-existent UET entry specified	:*
val	***$UECENV2***	000002	Duplicate UET entry specified	:*
val	***$UECENV3***	000003	UET memory overflow	:*
val	***$UECFIL***	000213	FILES,,FILES$	: D$ERRNUM
val	***$UECFILO***	000000	Catalog access denied	:*
val	***$UECFIL1***	000001	FILEPCN not called	:*
val	***$UECFIL2***	000002	Catalog file already closed	:*
val	***$UECFIO***	000204	FASTIO,Fast I/O	
val	***$UECFIOO***	000000	Input text file record too large	
val	***$UECFIO1***	000001	Invalid input text file format	
val	***$UECFIO2***	000002	Invalid char in text record to be written	
val	***$UECGLN***	000202	GETLINE,	: D$ERRNUM
val	***$UECGLNO***	000000	CHAIN file line too long	:*
val	***$UECGLN1***	000001	Keyin attempted with note display inhibitd	:*
val	***$UECGLN2***	000002	No workstation available	:*
val	***$UECLDR***	000212	LOADREL,	: D$ERRNUM
val	***$UECLDRO***	000000	Not enough memory to load member	:*
val	***$UECLDR1***	000001	Cannot allocate memory to load member	:*
val	***$UECLDR2***	000002	Library format error	:*
val	***$UECLDR3***	000003	Specified member could not be found	:*
val	***$UECLDR4***	000004	Invalid relocatable text control code	:*
val	***$UECLDR5***	000005	Invalid sector code encountered	:*
val	***$UECLDR6***	000006	Specified member is not correct type	:*
val	***$UECLDR7***	000007	Internal error	:*
val	***$UECLIB***	000214	LIBUFR,$UECLIB,0214,LBUFR$	: D$ERRNUM
val	***$UECLIBO***	000000	Member not found	:*
val	***$UECLIB1***	000001	Duplicate member	:*
val	***$UECLIB2***	000002	Invalid library file format	:*
val	***$UECLOG***	000211	Logging	: D$ERRNUM

val	*$UECLOG0*	000000 Invalid $WSIO control character encounterd:*
val	*$UECLOG1*	000001 Log file environment not found by ENVLOC$:*
val	*$UECLOG2*	000002 Log vector and flag are inconsistent :*
val	*$UECLOG3*	000003 Entry point error in PCR resident code :*
val	*$UECLOG4*	000004 Log data pointer and flag are inconsistent:*
val	*$UECLOG5*	000005 Invalid log message member format :*
val	*$UECLOG6*	000006 Internal error in log control program :*
val	*$UECLOG7*	000007 Invalid device type given for log output :*
val	*$UECMEM*	000215 MEMGP$: D$ERRNUM
val	*$UECMEM0*	000000 Attempt to exceed allowable memory :*
val	*$UECMP4K*	0207 $MAP4K : D$ERRNUM
val	*$UECNVD*	000217 ENVDEL$: D$ERRNUM
val	*$UECNVI*	000216 ENVINS$: D$ERRNUM
val	*$UECNVL*	000220 ENVLOC$: D$ERRNUM
val	*$UECNVP*	000221 ENVPEEL$: D$ERRNUM
val	*$UECOPN*	000201 OPEN,'File OPEN' : D$ERRNUM
val	*$UECOPN0*	000000 Specified environment is not defined :*
val	*$UECOPN1*	000001 File not found in any environment :*
val	*$UECOPN2*	000002 Invalid open mode for environment scanning:*
val	*$UECOPN3*	000003 Invalid file format code :*
val	*$UECPDA*	000206 DOSPDA,Physical Disk Address,NO DASL VAL?
val	*$UECPDA*	000206 not in FUNCTIONS Section .NO DASL VAL?
val	*$UECPDA0*	000000 Unknown disk sub-kind
val	*$UECPDA1*	000001 Invalid DOS physical disk address
val	*$UECSFL*	000205 SCANFLS, : D$ERRNUM
val	*$UECSFL0*	000000 Invalid character in file specification :*
val	*$UECSFL1*	000001 Undefined symbolic file name :*
val	*$UECSFL2*	000002 File specification duplicated :*
val	*$UECSFL3*	000003 Too many file specifications :*
val	*$UECSFL4*	000004 File prompt required while chaining or stack active :*
val	*$UECSFL5*	000005 File specification string too long :*
val	*$UECSOS*	000203 SCANOS, : D$ERRNUM
val	*$UECSOS0*	000000 Invalid character in option specification :*
val	*$UECSOS1*	000001 Undefined option name :*
val	*$UECSOS2*	000002 Option name duplicated :*
val	*$UECSOS3*	000003 Option specification string too long :*
val	*$UECSOS4*	000004 ?
ufr	*$UERMSG*	Store User Error Message on Command Stack :D$UFRERR
abr	*UET*	User Environment Table
var	*$UET1STP*	∧ BYTE; Adr of first byte in user ENV table : D$PCR
var	*$UETPTR*	∧ $ENVT; Address of first UET entry : D$PCR
var	*$UETTOPP*	∧ BYTE; Above top of UET area ? : D$PCR
abr	*UFR*	User Function Routine
typ	*ULONG*	STRUCT; 24 Bit Number Structure : D$INC
var	*$ULS1ST*	[0] BYTE; First byte of usr logical adr spac: D$PCR
var	*$ULSTOPP*	∧ BYTE; Above top of user logical space : D$PCR
ddw	*UNION*	Named Member Contains Different Possible Members
ufr	*$UNLKRIM*	Release RIM from Pipe ? (on hold) : D$UFRSYS
typ	*$UNPACKPW*	[8] CHAR; Unpacked Password : D$UFRENV
ufr	*$UNPAKPW*	Unpack password into ASCII string : D$UFRENV
ddt	*UNSIGNED*	;DASL scalar data type: 2 bytes unsigned
sc	*$USRABN*	User Abend Facility : D$RMSPROG
abr	*UTC*	Universal Time Co-ordinated (GMT)
val	*$UTEACCS*	1 User Tsk Err:Mem access protect violatn:D$RMSPROG

val	***$UTEHALT***	0377 User Tsk Err:Halt Ins for breakpntng	:D$RMSPROG
val	***$UTEINST***	2 Usr Tsk Err:Illegal Ins.usr mode violatn	:D$RMSPROG
val	***$UTEUNDF***	3 UsrTsk Err:Undefined Ins or system call	:D$RMSPROG
val	***$UTEWRIT***	0 UsrTsk Err:Memory write protect violatn	:D$RMSPROG
val	***$V***	0235 New cursor row follows (pos)	: D$RMS
val	***$VA***	0237 Cursor row adjustment follows (adj)	: D$RMSWS
ddw	***VAR***	Indicates Local Variable Definitions Follow	
var	***VBASEIO$***	EXTERN VBASEIO$ BYTE; ?	: D$UFRWFIO
ufr	***$VGETBUF***	Obtain Buffer Group from Virtual Pool	: D$UFRWFIO
ufr	***$VINIT***	Initialize Virtual I/O Management	: D$UFRWFIO
var	***VIOINIT$***	EXTERN VIOINIT$ BOOLEAN; ?	: D$UFRWFIO
var	***VIRTUAL$***	EXTERN VIRTUAL$ BYTE;	: D$UFRWFIO
	VIRTUAL$	Driver function for Virtual WFIO	
ufr	***$VMAPPSK***	Donate a PSK to Virtual Management	: D$UFRWFIO
abr	***VOLID***	Volume Identification	
ufr	***$VPUTBUF***	Return a Buffer Group to Virtual Pool	: D$UFRWFIO
abr	***VRP***	? in $INFOITEM	
ufr	***$VSETWIN***	Establish Memory Window Areas, Virtual	: D$UFRWFIO
sc	***$WAITIO***	Wait for any Operation Completion	: D$RMSIO
sc	***$WAITIOS***	Wait Status Change	: D$RMSIO
sc	***$WCONFIG***	Get Workstation Configuration	: D$RMSWS |*
typ	***$WFCB***	STRUCT; Work File I/O Control Block	: D$UFRWFIO
fld	***$WFCBBLKCOUNT***	UNSIGNED; block I/O counter	: $WFCB
fld	***$WFCBBLKCURR***	UNSIGNED; current block position	: $WFCB
fld	***$WFCBBLKMAX***	UNSIGNED; max block size allowed	: $WFCB
fld	***$WFCBBLKSIZE***	UNSIGNED; current block size	: $WFCB
fld	***$WFCBCURR***	$FILEPTR; Current file pointer	: $WFCB
fld	***$WFCBEOF***	$FILEPTR; EOF pointer	: $WFCB
fld	***$WFCBFLAG***	$WFCBFLAG; Control flag ($WFF..) :$WFCB	: D$UFRWFIO
typ	***$WFCBFLAG***	Work File I/O Flags	: D$UFRWFIO
fld	***$WFCBFLAG2***	$WFCBFLAG2; Second control flag byte	: $WFCB
typ	***$WFCBFLAG2***	Second Control Flag Byte Type	: D$UFRWFIO
fld	***$WFCBHOLD***	BYTE; Space compression hold area	: $WFCB
fld	***$WFCBMAX***	$FILEPTR; Maximum pointer	: $WFCB
fld	***$WFCBPFDBP***	^ $PFDB; PFDB pointer	: $WFCB
fld	***$WFCBPFDBP2***	^ $PFDB; Secondary PFDB pointer	: $WFCB
fld	***$WFCBPIO***	^ BYTE; Physical I/O routine vector	: $WFCB
fld	***$WFCBPIOTAB***	^ $WFIOTABPRTN; routines	: $WFCB
fld	***$WFCBRESV***	[4] BYTE; Reserved Area	: $WFCB
fld	***$WFCBRSIZ***	UNSIGNED; Record size	: $WFCB
ufr	***$WFCLOSE***	Terminate Processing of Work File	: D$UFRWFIO
val	***$WFFDIRTY***	002 Internal Use : $WFCBFLAG	: D$UFRWFIO
val	***$WFFEOFOK***	0200 ? : $WFCBFLAG	: D$UFRWFIO
val	***$WFFINPROG***	0100 ? : $WFCBFLAG	: D$UFRWFIO
ufr	***$WFFLUSH***	Dump Pending Write Buffers to Disk	: D$UFRWFIO
val	***$WFFNOTDSK***	040 Internal Use : $WFCBFLAG	: D$UFRWFIO
val	***$WFFSHRD***	020 Text or Binary File in Shared mode	
	$WFFSHRD	: $WFCBFLAG	: D$UFRWFIO
val	***$WFFSPCTXT***	004 Special text records to be read	
	$WFFSPCTXT	: $WFCBFLAG	: D$UFRWFIO
val	***$WFFUNCOMP***	010 Uncompressed format text: $WFCBFLAG	: D$UFRWFIO

```
val $WFFUPDEOF 001        ?                    : $WFCBFLAG : D$UFRWFIO
abr  WFIO      Work File I/O
typ $WFIOPRTN  Physical I/O Function Routine Type      : D$UFRWFIO
typ $WFIOTABPRTN Table of Pointers to Phys I/O Routines: D$UFRWFIO
ufr $WFOPEN    Prepare an Open Work File For Access    : D$UFRWFIO
ufr $WFOPENP   Open Using Specified Physical I/O Routine:D$UFRWFIO
val $WFPACKED  1    ?                     : $WFCBFLAG2 : D$UFRWFIO
var  WFPIO$    EXTERN WFPIO$ BYTE; Driver,             : D$UFRWFIO
     WFPIO$    Workfile Physical I/O Driver Routine
ufr $WFPOSEF   Position to Files EOF                   : D$UFRWFIO
ufr $WFPOSIT   Position To File Pointer                : D$UFRWFIO
ufr $WFPREP    Prepare a New Work File For Access      : D$UFRWFIO
ufr $WFPREPP   Prepare Using Specified Phys I/O Routine :D$UFRWFIO
ufr $WFREAD    Read a Logical Record                   : D$UFRWFIO
ufr $WFREADL   Read in LOCATE Mode                     : D$UFRWFIO
cr   WFUPDAT$  Update a Logical Record
ufr $WFUPDATE  Update a Logical Record                 : D$UFRWFIO
ufr $WFUPDATEL Update a Record in LOCATE Mode          : D$UFRWFIO
cr   WFUPDTL$  Update a Record in LOCATE Mode
ufr $WFWEOF    Write EOF At Current File Position      : D$UFRWFIO
ufr $WFWRITE   Write a Logical Record                  : D$UFRWFIO
ufr $WFWRITEL  Write a Record in LOCATE Mode           : D$UFRWFIO
cr   WFWRITL$  Write a Record in LOCATE Mode
dcw  WHILE     Part of LOOP WHILE Execution Control
ufr $WIPEBT    Clear an Area of Memory to SPACES        : D$UFRGEN
ufr $WIPEBTA   Clear an Area of Mem to Constant Value   : D$UFRGEN
abr  WPS       Word Processing System
val $WS0       1 Workstation Kind bit 0      : $WSCONF : D$RMSWS  |*
val $WS1       2 Workstation Kind bit 1      : $WSCONF : D$RMSWS  |*
val $WS2       4 Workstation Kind bit 2      : $WSCONF : D$RMSWS  |*
val $WS22LVA   8 2-Level video available    : $WSCONF2 : D$RMSWS
val $WS2BNKA   040 Blink available          : $WSCONF2 : D$RMSWS
val $WS2CFL    4 Cursor font loadable       : $WSCONF2 : D$RMSWS
val $WS2EFKO   0100 Expandd function kbd. on-line:$WSCONF2:D$RMSWS  |*
val $WS2IPL    2 System just Re-Booted      : $WSCONF2 : D$RMSWS
val $WS2POW    1 Tube just powered on       : $WSCONF2 : D$RMSWS
val $WS2ULNA   020 Underline available      : $WSCONF2 : D$RMSWS
val $WS3       010 Workstation Kind bit 3    : $WSCONF : D$RMSWS  |*
val $WS3FD0    0 Character Font Not Loadable  :$WSCONFDS : D$RMSWS
    $WS3FD1    thru $WS3FD4 : Table heading                       |*
    $WS3FD1    Size   Size  Length  Descriptor
    $WS3FD1    Horiz  Vert  Per Char Required?
val $WS3FD1    1,  5     7       5      no     : $WSCONFDS : D$RMSWS
val $WS3FD2    2,  5     7       7      no     : $WSCONFDS : D$RMSWS
val $WS3FD3    3,  8     12      12     yes    : $WSCONFDS : D$RMSWS
val $WS3FD4    4,  9     12      12     yes    : $WSCONFDS : D$RMSWS  |*
val $WS3FDMK   017 'Font Data' Mask            : $WSCONFDS : D$RMSWS
val $WS3NFMK   0360 'Number of Fonts' Mask     : $WSCONFDS : D$RMSWS
val $WS3NFS    4 'Number of Fonts' Shift Value:$WSCONFDS : D$RMSWS
val $WSALT1D   0334 'ALT (LEFT/LEFT)' key down             : D$RMSWS  |*
val $WSALT1U   0335 'ALT (LEFT/LEFT)' key up               : D$RMSWS  |*
val $WSALT2D   0310 'ALT (LEFT/RIGHT)' key down            : D$RMSWS  |*
val $WSALT2U   0311 'ALT (LEFT/RIGHT)' key up              : D$RMSWS  |*
val $WSALT3D   0312 'ALT (RIGHT/LEFT)' key down            : D$RMSWS  |*
val $WSALT3U   0313 'ALT (RIGHT/LEFT)' key up              : D$RMSWS  |*
val $WSALT4D    0336 'ALT (RIGHT/RIGHT)' key down          : D$RMSWS |*
val $WSALT4U    0337 'ALT (RIGHT/RIGHT)' key up            : D$RMSWS |*
val $WSALTLD    0334 'ALT (LEFT)' key down                 : D$RMSWS |*
val $WSALTLU    0335 'ALT (LEFT)' key up                   : D$RMSWS |*
val $WSALTRD    0336 'ALT (RIGHT)' key down                : D$RMSWS |*
val $WSALTRU    0337 'ALT (RIGHT)' key up                  : D$RMSWS |*
val $WSATT      02000 'ATTENTION' key Down       : $WSTAT : D$RMSWS
val $WSATTEN    0216 Enable KDS 3 Attributes               : D$RMSWS
    $WSATTEN    ( underline & 2-level video) on 8600 console.
    $WSATTEN    Has no effect on other workstations.
val $WSATTK     000217 'ATTENTION' key                     : D$RMSWS
val $WSATTKS    0231 'ATTENTION' key shifted               : D$RMSWS
val $WSATTUP    0222 'ATTENTION' key released              : D$RMSWS
val $WSBADPK    000220 Bad parity received                 : D$RMSWS
val $WSBAK1D    0302 'BACKSPACE (LEFT)' key down (U.S.A.)  : D$RMSWS |*
val $WSBAK1U    0303 'BACKSPACE (LEFT)' key up (U.S.A.)    : D$RMSWS |*
val $WSBAK2D    0316 'BACKSPACE (RIGHT)' key down (U.S.A.): D$RMSWS |*
val $WSBAK2U    0317 'BACKSPACE (RIGHT)' key up (U.S.A.)   : D$RMSWS |*
val $WSBAKSD    0316 'BACKSPACE' key down (U.S.A.)         : D$RMSWS |*
val $WSBAKSU    0317 'BACKSPACE' key up (U.S.A.)           : D$RMSWS |*
val $WSBCPA     040000 Cursor positioning available:$WSCONF:D$RMSWS
val $WSBEEP     0204 Beep                                    : D$RMS
val $WSBFKDA    02000 F1 thru F5 downstrokes avail:$WSCONF: D$RMSWS
val $WSBFKSA    04000 F1 thru F5 static bits avail:$WSCONF: D$RMSWS
val $WSBFKUA    01000 F1 thru F5 upstrokes avail  :$WSCONF: D$RMSWS
val $WSBFSHA    0400  Shifted function keys avail :$WSCONF: D$RMSWS
val $WSBIFDT    010   Get data from Internal Buffer Module: D$RMSWS |*
val $WSBIVA     020000 Inverted video available : $WSCONF : D$RMSWS
val $WSBL       000013 Bottom line                         : D$RMSWS
val $WSBLANK    040 ' ', Reserved for blank code           : D$RMSWS
val $WSBLCFA    010000 Display font set loadable : $WSCONF: D$RMSWS
val $WSBSPK     000010 'BACKSPACE' key                     : D$RMSWS
val $WSBSWA     0100000 Sub windows available    : $WSCONF : D$RMSWS
val $WSCANK     000030 'CANCEL' key                        : D$RMSWS
val $WSCASED    0326 'CASE INVERT' key down                : D$RMSWS |*
val $WSCASEU    0327 'CASE INVERT' key up                  : D$RMSWS |*
val $WSCCKA     000200 Click available           : $WSCONF : D$RMSWS
val $WSCIAKA    000020 INT & ATT keys downstrokes avail.   : D$RMSWS
val $WSCIAUA    0100 was $WSS6, and $WSCVRWA     : $WSCONF : D$RMSWS
val $WSCIAUA    0100 INTERRUPT & ATTENTION keys static bits and
    $WSCIAUA    upstrokes available             :$WSCONF : D$RMSWS
val $WSCIRCD    0250 'CIRCLE' key down                     : D$RMSWS |*
val $WSCIRCU    0251 'CIRCLE' key up                       : D$RMSWS |*
val $WSCKCCD    0262 Cursor Key (CENTER-CENTER) down       : D$RMSWS |*
val $WSCKCCU    0263 Cursor Key (CENTER-CENTER) up         : D$RMSWS |*
val $WSCKCLD    0260 Cursor Key (CENTER-LEFT) down         : D$RMSWS |*
val $WSCKCLU    0261 Cursor Key (CENTER-LEFT) up           : D$RMSWS |*
val $WSCKCRD    0264 Cursor Key (CENTER-RIGHT) down        : D$RMSWS |*
val $WSCKCRU    0265 Cursor Key (CENTER-RIGHT) up          : D$RMSWS |*
val $WSCKDCD    0270 Cursor Key (DOWN-CENTER) down         : D$RMSWS |*
val $WSCKDCU    0271 Cursor Key (DOWN-CENTER) up           : D$RMSWS |*
val $WSCKDKA    KBD & DPY keys static bits & upstrokes avail
    $WSCKDKA    000040                           : $WSCONF : D$RMSWS
val $WSCKDLD    0266 Cursor Key (DOWN-LEFT) down           : D$RMSWS |*
val $WSCKDLU    0267 Cursor Key (DOWN-LEFT) up             : D$RMSWS |*
```

val	***$WSCKDRD***	0272 Cursor Key (DOWN-RIGHT) down	: D$RMSWS	|*
val	***$WSCKDRU***	0273 Cursor Key (DOWN-RIGHT) up	: D$RMSWS	|*
val	***$WSCKF***	0247 Clear keyboard fifo	: D$RMSWS	
val	***$WSCKUCD***	0254 Cursor Key (UP-CENTER) down	: D$RMSWS	|*
val	***$WSCKUCU***	0255 Cursor Key (UP-CENTER) up	: D$RMSWS	|*
val	***$WSCKULD***	0252 Cursor Key (UP-LEFT) down	: D$RMSWS	|*
val	***$WSCKULU***	0253 Cursor Key (UP-LEFT) up	: D$RMSWS	|*
val	***$WSCKURD***	0256 Cursor Key (UP-RIGHT) down	: D$RMSWS	|*
val	***$WSCKURU***	0257 Cursor Key (UP-RIGHT) up	: D$RMSWS	|*
val	***$WSCLICK***	0205 Click	: D$RMS	
val	***$WSCLOSL***	0005 Close line from under cursor rolling	: D$RMSWS	
val	***$WSCMDD***	0274 'COMMAND' key down	: D$RMSWS	|*
val	***$WSCMDU***	0275 'COMMAND' key up	: D$RMSWS	|*
val	***$WSCMODE***	0246 Clear mode (bits)	: D$RMSWS	
fld	***$WSCON2***	$WSCONF2; ? : $WSCONFDS	: D$RMSWS	
fld	***$WSCON3***	BYTE; ? : $WSCONFDS	: D$RMSWS	
fld	***$WSCONC***	$WSCONF; ? : $WSCONFDS	: D$RMSWS	
typ	***$WSCONF***	$WCONFIG Status Bits	: D$RMSWS	
typ	***$WSCONF2***	$WCONFIG third status byte	: D$RMSWS	|*
val	***$WSCONFD***	0272 WS Config data (Len),((Loc))	: D$RMSWS	
typ	***$WSCONFDS***	$WCONFIG 4 byte status structure	: D$RMSWS	
val	***$WSCOPYD***	0214 'COPY' key down	: D$RMSWS	|*
val	***$WSCOPYU***	0215 'COPY' key up	: D$RMSWS	|*
val	***$WSCR***	0267 Carriage return (WS serial printers)	: D$RMSWS	
sc	***$WSCTL***	Workstation Control Code Function	: D$RMSWS	
val	***$WSCTLBP***	0003 Beep	: D$RMSWS	
val	***$WSCTLCF***	0000 Cursor off	: D$RMSWS	
val	***$WSCTLCK***	0002 Click	: D$RMSWS	
val	***$WSCTLCN***	0001 Cursor on	: D$RMSWS	
val	***$WSCURDF***	0275 Return to default cursor font	: D$RMSWS	
val	***$WSCURFL***	0274 Load cursor font from ((Loc))	: D$RMSWS	
val	***$WSCUROF***	0215 Turn Cursor Off at current position	: D$RMSWS	
val	***$WSCURON***	0214 Turn Cursor On at current position	: D$RMSWS	
val	***$WSCURS***	0000 Reserved for cursor code (1800)	: D$RMSWS	
val	***$WSDELCH***	0001 Delete char under cursor, shift up	: D$RMSWS	
val	***$WSDELK***	0177 'DEL' key (used by katakana)	: D$RMSWS	
val	***$WSDELLN***	0003 Delete line under cursor and roll up	: D$RMSWS	
val	***$WSDKPOF***	004 Disable dead key processor	: D$RMSWS	|*
val	***$WSDLLR***	0235 'DLL response (8220)' character	: D$RMSWS	|*
val	***$WSDSCNT***	0 Disconnect datastation	: D$RMSWS	|*
val	***$WSDSP***	0040 'DISPLAY' key down : $WSTAT	: D$RMSWS	
val	***$WSDSPK***	0200 'DISPLAY' key	: D$RMSWS	
val	***$WSDSPKS***	0232 'DISPLAY' key shifted	: D$RMSWS	
val	***$WSDSPUP***	0201 'DISPLAY' key released	: D$RMSWS	
val	***$WSECHO***	0052 '*' used in keyin echo	: D$RMSWS	
val	***$WSECHOS***	0262 Set echo secret disply char (char)	: D$RMSWS	
val	***$WSEFBPK***	0342 Bad parity keycode - E.F. keyboard	: D$RMSWS	|*
val	***$WSEFCL***	016 $WSEFST2; Last EFK Control code value	: D$RMSWS	|*
val	***$WSEFCTL***	1 expanded function keyboard control	: D$RMSWS	|*
val	***$WSEFCTT***	0337 E.F. keyboard - top control value	: D$RMSWS	|*
val	***$WSEFDCI***	6 disable case inversn;Keycode Xlate Module	:D$RMSWS	|*
val	***$WSEFECI***	5 enable case inversn; Keycode Xlate Module	:D$RMSWS	|*
val	***$WSEFF1D***	0230 function key '1' down	: D$RMSWS	|*
val	***$WSEFF1U***	0231 function key '1' up	: D$RMSWS	|*
val	***$WSEFF2D***	0232 function key '2' down	: D$RMSWS	|*

```
val $WSEFF2U    0233 function key '2' up                : D$RMSWS  |*
val $WSEFF3D    0234 function key '3' down              : D$RMSWS  |*
val $WSEFF3U    0235 function key '3' up                : D$RMSWS  |*
val $WSEFF4D    0236 function key '4' down              : D$RMSWS  |*
val $WSEFF4U    0237 function key '4' up                : D$RMSWS  |*
val $WSEFF5D    0240 function key '5' down              : D$RMSWS  |*
val $WSEFF5U    0241 function key '5' up                : D$RMSWS  |*
val $WSEFF6D    0242 function key '6' down              : D$RMSWS  |*
val $WSEFF6U    0243 function key '6' up                : D$RMSWS  |*
val $WSEFF7D    0224 function key '7' down              : D$RMSWS  |*
val $WSEFF7U    0225 function key '7' up                : D$RMSWS  |*
val $WSEFF8D    0226 function key '8' down              : D$RMSWS  |*
val $WSEFF8U    0227 function key '8' up                : D$RMSWS  |*
val $WSEFKAB    012  Set field keyin abort (KEY);       : D$RMSWS  |*
val $WSEFKAR    013  Reset all field keyin abort keys;  : D$RMSWS  |*
val $WSEFKFO    0341 extended function keyin FIFO overflow: D$RMSWS |*
val $WSEFKID    3    Get keyboard I.D. ((LOC))          : D$RMSWS  |*
val $WSEFKTP    014  Activate function trap for key (KEY);: D$RMSWS |*
    $WSEFKTP         (Keycode Translate Module)                    |*
val $WSEFKTR    015  Reset all function key traps;      : D$RMSWS  |*
val $WSEFLCD    2    LCD control (MASK),(BYTE);         : D$RMSWS  |*
val $WSEFLOW    1    Expandd function KBD Flow control(CTL):D$RMSWS |*
val $WSEFONL    0200 'ONLINE' status (always true)      : D$RMSWS  |*
val $WSEFRDY    0001 Key ready                          : D$RMSWS  |*
val $WSEFRPT    4    set repeat key timeout (TIME)      : D$RMSWS  |*
    $WSEFRPT         (Keycode Translate Module)                    |*
val $WSEFRST    0  Reset all but Keycode Xlate Module path:D$RMSWS |*
val $WSEFST2    016 Get status bits + $WSTATUS bits ((LOC)):D$RMSWS |*
val $WSEFSTC    7    Get current static bits ((LOC))    : D$RMSWS  |*
val $WSEFSTF    010  Get static fifo bits ((LOC))       : D$RMSWS  |*
val $WSEFSTL    011  Get latched static bits ((LOC))    : D$RMSWS  |*
val $WSENT1D    0276 'ENTER (LEFT)' key down (U.S.A.)   : D$RMSWS  |*
val $WSENT1U    0277 'ENTER (LEFT)' key up (U.S.A.)     : D$RMSWS  |*
val $WSENT2D    0314 'ENTER (RIGHT)' key down (U.S.A.)  : D$RMSWS  |*
val $WSENT2U    0315 'ENTER (RIGHT)' key up (U.S.A.)    : D$RMSWS  |*
val $WSENTD     0314 $WSENT2D  'ENTER' key down (U.S.A.) : D$RMSWS |*
val $WSENTK     0015 'ENTER' key                          : D$RMS
val $WSENTU     0315 $WSENT2U  'ENTER' key up (U.S.A.)  : D$RMSWS  |*
val $WSESC1     0226 WSIO 'escape-sequence-1' codes follow: D$RMSWS |*
val $WSESC1L    1   $WSEFCTL;last escape-seq-1 control code:D$RMSWS |*
val $WSF1       0001 'F1' key down            : $WSTAT : D$RMSWS
val $WSF1K      0204 'F1' key                          : D$RMSWS
val $WSF1KS     0223 'F1' key shifted                  : D$RMSWS
val $WSF1UP     0205 'F1' key released                 : D$RMSWS
val $WSF2       0002 'F2' key down            : $WSTAT : D$RMSWS
val $WSF2K      0206 'F2' key                          : D$RMSWS
val $WSF2KS     0224 'F2' key shifted                  : D$RMSWS
val $WSF2UP     0207 'F2' key released                 : D$RMSWS
val $WSF3       0004 'F3' key down            : $WSTAT : D$RMSWS
val $WSF3K      0210 'F3' key                          : D$RMSWS
val $WSF3KS     0225 'F3' key shifted                  : D$RMSWS
val $WSF3UP     0211 'F3' key released                 : D$RMSWS
val $WSF4       0010 'F4' key down            : $WSTAT : D$RMSWS
val $WSF4K      0212 'F4' key                          : D$RMSWS
val $WSF4KS     0226 'F4' key shifted                  : D$RMSWS
```

val	***$WSF4UP***	0213 'F4' key released	: D$RMSWS	
val	***$WSF5***	0020 'F5' key down : $WSTAT	: D$RMSWS	
val	***$WSF5K***	0214 'F5' key	: D$RMSWS	
val	***$WSF5KS***	0227 'F5' key shifted	: D$RMSWS	
val	***$WSF5UP***	0215 'F5' key released	: D$RMSWS	
val	***$WSFF***	0266 Form feed (for WS serial printers)	: D$RMSWS	
val	***$WSGENUP***	0340 generic rollover key upstroke	: D$RMSWS	|*
sc	***$WSGETCH***	Obtain One Keyboard Buffer Character	: D$RMSWS	
val	***$WSHELPD***	0212 'HELP' key down	: D$RMSWS	|*
val	***$WSHELPU***	0213 'HELP' key up	: D$RMSWS	|*
val	***$WSIDOCS***	0224 INS, DEL, OPEN or CLOSE follows	: D$RMSWS	
val	***$WSIKCOF***	0211 Key click off	: D$RMSWS	
val	***$WSIKCON***	0210 Key click on	: D$RMSWS	
val	***$WSIN***	0254 In numeric (lmax),(rmax),((loc)),(end)	:D$RMSWS	
val	***$WSINI***	0255 In numrc imm(lmax),(rmax),(skip),(end)	:D$RMSWS	
val	***$WSINSCH***	0000 Insert space under cursor, shift down	:D$RMSWS	
val	***$WSINSLN***	0002 Roll down lines from cursor to bottom	:D$RMSWS	
val	***$WSINSTD***	0220 'INSERT' key down	:D$RMSWS	|*
val	***$WSINSTU***	0221 'INSERT' key up	:D$RMSWS	|*
val	***$WSINT***	01000 'INTERRUPT' key down : $WSTAT	: D$RMSWS	
val	***$WSINTK***	0216 'INTERRUPT' key	: D$RMSWS	
val	***$WSINTKS***	0230 'INTERRUPT' key shifted	: D$RMSWS	
val	***$WSINTUP***	0221 'INTERRUPT' key released	: D$RMSWS	
sc	***$WSIO***	Perform Workstation I/O	: D$RMSWS	
val	***$WSIOFCF***	0200 $WSIO function code first value	: D$RMSWS	
val	***$WSIOFCL***	0276 $WSK1CHR; $WSIO func code last value	: D$RMSWS	|*
typ	***$WSIOMODE***	$WSIO Mode Bits	: D$RMSWS	
val	***$WSIS***	0252 In string (con),(max),((loc)),(end)	: D$RMSWS	
val	***$WSISI***	0253 In string imm (con),(max),(skip),(end)	:D$RMSWS	
val	***$WSITIME***	0256 Set inter-char timeout to (t) seconds	: D$RMSWS	
val	***$WSK1CHR***	0276 Keyin un-xlated char at ((LOC))	: D$RMSWS	|*
	$WSK1CHR	with cursor on/off		|*
val	***$WSKBD***	0100 'KEYBOARD' key down : $WSTAT	: D$RMSWS	
val	***$WSKBDK***	0202 'KEYBOARD' key	: D$RMSWS	
val	***$WSKBDKS***	0233 'KEYBOARD' key shifted	: D$RMSWS	
val	***$WSKBDUP***	0203 'KEYBOARD' key released	: D$RMSWS	
val	***$WSKCTLT***	0235 $WSKDLLR; Kbd control code top value	: D$RMSWS	|*
val	***$WSKEYCH***	0270 Keyin un-xlated character at ((Loc))	: D$RMSWS	
val	***$WSKFULL***	0234 'Keyin FIFO Full' Character	: D$RMSWS	
val	***$WSKMASK***	0017 Workstation kind in bits 0-3	: D$RMSWS	
val	***$WSLC***	0000 Left column	: D$RMSWS	
val	***$WSLCDCR***	0100 CIRCLE (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDDP***	0004 DISPLAY (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDKD***	0010 KEYBOARD (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDLC***	0001 LOWER CASE (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDSQ***	0020 SQUARE (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDTR***	0040 TRIANGLE (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCDUC***	0002 UPPER CASE (0/1) (OFF/ON)	: D$RMSWS	|*
val	***$WSLCFS***	0260 Load char font set from ((loc))	: D$RMSWS	
val	***$WSLENON***	001 enable Keycode Xlate Module path	: D$RMSWS	|*
val	***$WSLF***	0265 Line feed (for WS serial printers)	: D$RMSWS	
val	***$WSLNTOF***	020 disable Keycode Translate Module path	:D$RMSWS	|*
val	***$WSLOCKD***	0326 'LOCK' key down	: D$RMSWS	|*
val	***$WSLOCKU***	0327 'LOCK' key up	: D$RMSWS	|*
val	***$WSLSTSK***	0222 $WSATTUP, Last static key code	: D$RMSWS	

```
val $WSMDIGO   0040 Digits only              : $WSIOMODE : D$RMSWS
val $WSMES     0002 Echo secret (char)       : $WSIOMODE : D$RMSWS
val $WSMKCON   0020 Keyin continuous         : $WSIOMODE : D$RMSWS
val $WSMNE     0010 No echo or cursor        : $WSIOMODE : D$RMSWS
val $WSMNESC   0200 No escape b4 0-037 or 0177:$WSIOMODE : D$RMSWS
val $WSMNI     0004 No case inversion        : $WSIOMODE : D$RMSWS
val $WSMNW     0001 Inhibit 'DISPLAY' key wait:$WSIOMODE : D$RMSWS
val $WSMPADN   0100 Pad numeric decimal part : $WSIOMODE : D$RMSWS
val $WSNOP     0244 No operation                         : D$RMSWS
val $WSNPE1D   0300 number pad 'ENTER' key (top) down    : D$RMSWS |*
val $WSNPE1U   0301 number pad 'ENTER' key (top) up      : D$RMSWS |*
val $WSNPE2D   0324 number pad 'ENTER' key (bottom) down : D$RMSWS |*
val $WSNPE2U   0325 number pad 'ENTER' key (bottom) up   : D$RMSWS |*
val $WSNPEND   0324 number pad 'ENTER' key down          : D$RMSWS |*
val $WSNPENU   0325 number pad 'ENTER' key up            : D$RMSWS |*
val $WSNPTBD   0322 number pad 'TAB' key down            : D$RMSWS |*
val $WSNPTBU   0323 number pad 'TAB' key up              : D$RMSWS |*
val $WSONCH    0251 Output repeated (char),(n) times     : D$RMSWS
val $WSONL     0200 'ONLINE' status (always true):$WSTAT : D$RMSWS
val $WSOPENL   0004 Open line from under cursor rolling  : D$RMSWS
val $WSOS      0250 Output string (len),((loc))          : D$RMSWS
val $WSPNTD    0206 POINT' key down                      : D$RMSWS |*
val $WSPNTU    0207 POINT' key up                        : D$RMSWS |*
val $WSPTOFF   0213 Turn Off Printer connected to WS     : D$RMSWS
val $WSPTRON   0212 Turn On Printer connected to WS      : D$RMSWS
val $WSPUPOF   002  Disable Locked Key Processing        : D$RMSWS |*
val $WSQUITD   0204 'QUIT' key down                      : D$RMSWS |*
val $WSQUITU   0205 'QUIT' key up                        : D$RMSWS |*
val $WSRC      0117 Right column                         : D$RMSWS
val $WSRDY     0400 Key ready                  : $WSTAT  : D$RMSWS
val $WSRECLD   0222 'RECALL' key down                    : D$RMSWS |*
val $WSRECLU   0223 'RECALL' key up                      : D$RMSWS |*
val $WSRECON   0273 WS Reconfig data (Len),((Loc))       : D$RMSWS
    $WSRECON          where ((Loc)) has the format:
    $WSRECON          Mask_0, Value_0, Mask_1, Value_1
    $WSRECON          Mask_(Len-1), Value_(Len-1)
val $WSREMVD   0216 'REMOVE' key down                    : D$RMSWS |*
val $WSREMVU   0217 'REMOVE' key up                      : D$RMSWS |*
val $WSRESET   0217 Reset Window to Default Screen size  : D$RMSWS
val $WSRESTR   0225 Same as $WSRESET except              : D$RMSWS
    $WSRESTR          8600 KDS attributes are not disabled
val $WSRPTK    0375 Repeated key;returnd only by $WSKEYCH :D$RMSWS
val $WSSCRE    0010 End of scroll data                   : D$RMSWS
val $WSSCRL    0006 Scroll left < followed by data >     : D$RMSWS
val $WSSCRR    0007 Scroll right < followed by data >    : D$RMSWS
val $WSSHF1D   0330 'SHIFT (LEFT/LEFT)' key down (U.S.A.): D$RMSWS |*
val $WSSHF1U   0331 'SHIFT (LEFT/LEFT)' key up (U.S.A.)  : D$RMSWS |*
val $WSSHF2D   0304 'SHIFT (LEFT/RIGHT)' key down (U.S.A.: D$RMSWS |*
val $WSSHF2U   0305 'SHIFT (LEFT/RIGHT)' key up (U.S.A.) : D$RMSWS |*
val $WSSHF3D   0332 'SHIFT (RIGHT/LEFT)' key down (U.S.A.: D$RMSWS |*
val $WSSHF3U   0333 'SHIFT (RIGHT/LEFT)' key up (U.S.A.) : D$RMSWS |*
val $WSSHF4D   0306 'SHIFT (RIGHT/RIGHT)' key down (U.S.A: D$RMSWS |*
val $WSSHF4U   0307 'SHIFT (RIGHT/RIGHT)' key up (U.S.A.): D$RMSWS |*
val $WSSHFLD   0330 'SHIFT (LEFT)' key down (U.S.A.)     : D$RMSWS |*
val $WSSHFLU   0331 'SHIFT (LEFT)' key up (U.S.A.)       : D$RMSWS |*
```

val	*$WSSHFRD*	0332 'SHIFT (RIGHT)' key down (U.S.A.)		: D$RMSWS	|*
val	*$WSSHFRU*	0333 'SHIFT (RIGHT)' key up (U.S.A.)		: D$RMSWS	|*
val	*$WSSHFTF*	0330 first SHIFT key		: D$RMSWS	|*
val	*$WSSHFTL*	0337 $WSALTRU, last SHIFT key		: D$RMSWS	|*
	$WSSHSW	000221 *** being phased out *** NOT IN DASL			
val	*$WSSKXTA*	0257 Set keyin translate table at ((loc))		: D$RMSWS	
val	*$WSSKXTP*	0261 Set keyin txlate table at((loc)),(psk)		:D$RMSWS	|*
val	*$WSSMODE*	0245 Set mode (bits)		: D$RMSWS	
val	*$WSSQARD*	0244 'SQUARE' key down		: D$RMSWS	|*
val	*$WSSQARU*	0245 'SQUARE' key up		: D$RMSWS	|*
val	*$WSSVMOD*	0264 Set video mode (mode)		: D$RMSWS	
	$WSSVRW	0220 *** being phased out *** NOT IN DASL			
val	*$WSSWLR*	0223 Set sub window (horz-left),(horz-rt)		: D$RMSWS	
val	*$WSSWTB*	0222 Set sub window (vert-top),(vert-bot)		: D$RMSWS	
val	*$WSSYSD*	0200 'SYSTEM' key down		: D$RMSWS	|*
val	*$WSSYSU*	0201 'SYSTEM' key up		: D$RMSWS	|*
val	*$WSTABD*	0320 'TAB' key down		: D$RMSWS	|*
val	*$WSTABU*	0321 'TAB' key up		: D$RMSWS	|*
typ	*$WSTAT*	$WSTATUS Status Bits		: D$RMSWS	
sc	*$WSTATUS*	Workstation, Get Status		: D$RMSWS	
val	*$WSTIMEO*	0376 Keyin timeout abort char;Aborts $WSIO		:D$RMSWS	
val	*$WSTRIAD*	0246 'TRIANGLE' key down		: D$RMSWS	|*
val	*$WSTRIAU*	0247 'TRIANGLE' key up		: D$RMSWS	|*
val	*$WSTWAIT*	0263 Perform n second wait (n)		: D$RMSWS	
val	*$WSUNDOD*	0210 'UNDO' key down		: D$RMSWS	|*
val	*$WSUNDOU*	0211 'UNDO' key up		: D$RMSWS	|*
val	*$WSVI*	0206 Video inverted		: D$RMSWS	
val	*$WSVIEWD*	0202 'VIEW' key down		: D$RMSWS	|*
val	*$WSVIEWU*	0203 'VIEW' key up		: D$RMSWS	|*
val	*$WSVM2L*	0000 Vid Mode: Bold-face, double intensity		:D$RMSWS	
val	*$WSVMAF*	0003 Video Mode: Alternate font		: D$RMSWS	
val	*$WSVMBNK*	0002 Video Mode: Blink		: D$RMSWS	
val	*$WSVMUNL*	0001 Video Mode: Underline		: D$RMSWS	
val	*$WSVN*	0207 Video normal		: D$RMSWS	
sc	*$WSWAIT*	Enable Port and Wait for Character		: D$RMSWS	
val	*$WSWAKEK*	0377 Wake up; Aborts $WSIO		: D$RMSWS	
fld	*$X1*	[4-1] $PFDBBUF;		: $ACB	
fld	*$X1*	[8-1] $PFDBBUF; 7 more buffers		: $ICB	
fld	*$X2*	[8-1] $PFDBBUF;		: $ACB	
fld	*$X7*	[16-1] $PFDBBUF; 15 more buffers		: $DCB	
val	*$XCFMS00*	000 Dummy Code, Undefined	: $SCFMSCODES	: D$FAR	
val	*$XCFMS01*	001 File pos out of range	: $SCFMSCODES	: D$FAR	
val	*$XCFMS02*	002 No such record	: $SCFMSCODES	: D$FAR	
val	*$XCFMS03*	003 ISAM key not found	: $SCFMSCODES	: D$FAR	
val	*$XCFMS04*	004 ISAM duplicate key	: $SCFMSCODES	: D$FAR	
val	*$XCFMS05*	005 Record already exists	: $SCFMSCODES	: D$FAR	
val	*$XCFMS06*	006 No current record exists	: $SCFMSCODES	: D$FAR	
val	*$XCFMS07*	007 File positioned to EOF	: $SCFMSCODES	: D$FAR	
val	*$YES*	000131 'Y', DEFINE'd		: D$PCR	

102706852