#### **General Disclaimer**

### One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some
  of the material. However, it is the best reproduction available from the original
  submission.

Produced by the NASA Center for Aerospace Information (CASI)

JM6/ Tech Cib.

# HAL/S-FC COMPILER SYSTEM SPECIFICATION

IR-95-5

1 March 1976

NASA CR. 147564

(NASA-CR-147564) HAL/S-FC COMPILER SYSTEM SPECIFICATIONS (Intermetrics, Inc.) 387 p CSCL 098

N76-22926

Unclas G3/60 25972

INTERMETRICS APPROVAL

Daniel J. Lickly

Head, HAL Language Compiler Department

Dr. F. H. Martin

Shuttle Program Manager





INTERMETRICS

### Table of Contents

1-1
1-1 1-1 1-2
2-1
2-2 JUDE
2-3 2-4 2-6 2-6 2-8
2-8 2-9 2-10 2-11 Table
eference 2-11 2-12 2-12
2-12 2-13 2-13 2-14 2-14
<b>3-1</b>
3-1
3-1 3-18 3-24 3-29 3-32 3-46 3-52 3-54 3-55 3-64 3-70 3-74 3-80 3-81

		Table	of Contents (Continued)	
				Page
	3.2 3.3 3.4 3.5	Printed Symbol	Code Naming Conventions d Data From Phase II Table Augmentation ent Table Augmentation	3-83 3-84 3-84 3-85
4.0	PHAS	E III -	SIMULATION DATA FILE GENERATION	4-1
	4.1	SDF Ger	neration	4-1
		4.1.1	Overall SDF Design	4-1
	4.2	Phase :	III Printed Data	4-3
5.0	RUN '	rime Lii	BRARY	5-1
		Introdu		5-1
	5.2	Basics	and Conventions	5-1
		5.2.2 5.2.3		5-1 5-1 5-2
		5.2.4	Register Conventions in Run Time Library Routines	5-2
		5.2.5	Referencing Conventions	5-10
		5.2.6	Coding Structure The Macro Library	5-11 5-12
	5,3	Librar	y Routine Descriptions	5-31
		5.3.1		5-52
			Algebraic Routine Description	5-71
		5.3.3	Vector/Matrix Routine Descriptions	5-135
		5.3.4	Character Routine Descriptions	5-24
		5.3.5	Array Function Routine Descrip-	E 260
		5.3.6	tions Miscellaneous Routine Descriptions	5-268 5-289
		5.3.7	REMOTE Routine Descriptions	5-33
6.0	SYST	EM INTE	RFACES	6-1
	6.1	Interna	al System Interfaces	6-1
		6.1.1	Macro Instructions	6-1
		6.1.2	Dynamic Invocation of the Compiler	6-2
		6.1.3	OS/360 Access Methods	6-2

## Table of Contents (Continued)

	Page
6.2 User or External System Interfaces	6-3
6.2.1 User-defined Options 6.2.2 Job Control Language Specification	6-3 6-3
Appendices:	
Appendix A: Compile-Time JCL Options Appendix B: Compiler Directives Appendix C: Error Classifications	A-1 B-1 C-1

#### 1.0 INTRODUCTION

### 1.1 Scope of Document

This document specifies the informational interfaces within the HAL/S-FC compiler, and between the compiler and the external environment. An overall description of the compiler, and the hardware and software compatibility requirements between compiler and environment are detailed in the HAL/S-FC Compiler Functional Specification<sup>1</sup>. Familiarization with the Functional Specification is presumed throughout this document.

This Compiler System Specification is for the HAL/S-FC compiler and its associated run time facilities which implement the full HAL/S language<sup>2</sup>. The HAL/S-FC compiler is designed to operate "stand-alone" on any compatible IBM 360/370 computer and within the Software Development Laboratory (SDL) at NASA/JSC, Houston, Texas.

### 1.2 Outline of the Document

The HAL/S-FC compiler system consists of:

- 1) a four phase language processor (compiler) which produces object modules compatible with AP-101 Space Shuttle Support Software and a set of simulation tables to aid in run time verification.
- 2) a comprehensive run-time library which provides an extensive set of mathematical, conversion, and language support routines.

The organization of this document is based upon the organization of the compiler system. Each part of the system is considered as a separate entity with its own specific function and interfaces to other parts. Hence, there are four sections which cover the parts of the system as follows:

HAL/S-FC Compiler System Functional Specification, 24 July 1974, IR #59-4.

<sup>2</sup> HAL/S Language Specification, 14 November 1975, IR #61-7.

- Section 2 describes Phase I and the syntax analysis phase of the compiler.
- Section 3 describes Phase II and the code generation phase and specifies in detail the code patterns for specific HAL/S constructs.
- Section 4 describes Phase III and the operation of the Simulation Data File generator.
- Section 5 describes the Runtime Library and the concepts used in the library and also gives specific information about each library routine including size, speed, and algorithm.

In addition to this part-by-part documentation, the compiler system, taken as a whole, exhibits properties and interfaces which are not specific to any one of the pieces. General information about such topics as the compiler's operating environment and user-written interfaces to emitted object code are contained in Section 6. Several Appendices are included which deal with tabular data used in the compiler system.

### 1.3 Status of Document

This document, plus the HAL/S-FC Compiler System Functional Specification comprise the complete HAL/S-FC Compiler System Specification. This publication is a specification for Release 10.0 of the HAL/S-FC compiler system.

The HAL/S-FC compiler inherits some of its operational features from the HAL/S-360 compiler system for which a similar Specification exists. In addition, many features of the HAL/S-FC system are under control of Interface Control Documents which are subject to update. When appropriate within this document, references are made to these companion documents as sources of supplementary material and in some cases as primary sources of detailed information.

The following list of documents represents the set of additional documents which reflect design and control of the HAL/S-FC compiler system:

- HAL/S-FC Compiler System Functional Specification, IR #59-4, 24 July 1974, by Intermetrics, Inc.
- Interface Control Document: HAL/FCOS, Revision 3, Published by IBM Federal Systems Division, Houston, Texas.
- Interface Control Document: HAL/SDL, Revision 6, Published by IBM Federal Systems Division, Houston, Texas.
- HAL/S-360 Compiler System Specification, IR #60-4, by Intermetrics, Inc.
- HAL/S Language Specification, IR #61-7, Published by Intermetrics, Inc.

### 2.0 PHASE I - SYNTAX ANALYSIS

The Syntax Analysis Phase performs syntactic and semantic analysis of the user's HAL/S source programs. It performs all functions necessary to allow an independent Phase II program to generate code for the target computer. The basic design of the HAL/S system includes use of a single Phase I for a variety of target machine Phase II's. Thus, the Phase I used by the HAL/S-FC compiler is the same one used in the HAL/S-360 compiler. In this section on Phase I, data which is supplied in detail in the HAL/S-360 Compiler System Specification is not repeated. Instead, reference is made to the proper section of that document.

This section deals with the following Phase I functions:

- Primary Source Input
- Secondary Source Input
- ACCESS System Implementation
- Compiler Directives
- Template Checking and Generation
- Printed Data
- Symbol Table Creation
- Statement Table Generation
- Literal Table Generation
- HALMAT Creation
- The Optimizer

### 2.1 Primary Source Input

Phase I accepts primary source input in the form of fixed length logical records. This input must be defined by the SYSIN DD statement in the JCL invoking the compiler. The first byte of each record is used to define the type of the record as follows:

- M main line
- E exponent line
- s subscript line
- D compiler directive
- C comment

For stand-alone operation the source records are 80 bytes in length and may contain data in columns 2-80. Optionally, the user may designate, via the "SRN" compiler option, that the source scanning is to stop at position 72 and also that positions 73-78 are to be printed on the listing as "Statement Reference Numbers".

When operating in the SDL environment, indicated by use of the "SDL" compiler option, the source records must still be all the same length but that length may be from 80 to 132 characters. When in the SDL mode, the compiler accepts source data from record positions 2 through 72. In addition, when the records are of sufficient length, the following fields are recognized and printed on the primary source listing:

- Record Sequence Number positions 73 through 78;
- Record Revision Indicator positions 79 and 80;
- Change Authorization Field positions 81 88.

REPRODUCIBILITY OF THE ORIGINAL PAGE 15 FOOT

Portions of records beyond position 88 are ignored.

The compiler's primary input may optionally be in a compressed source format as defined in the HAL/SDL ICD. No special notification of use of compressed source is needed. Phase I determines the type of input by examining the first record. Catonated datasets defined as primary compiler input must all be either in compressed or non-compressed format for one invocation of the compiler.

### 2.2 Secondary Source Input - The INCLUDE System

The user may direct the compiler to an alternate input source by use of an INCLUDE compiler directive in the primary input. The exact form of the INCLUDE directive may be found in Appendix B.

The INCLUDE directive defines a member name in a partitioned dataset. Phase I uses a FIND macro to locate the member on the INCLUDE DD card. If the FIND is unsuccessful, an identical FIND is issued for the OUTPUT6 DD card. A member, when located, is read to its end by the compiler. The records are processed identically to primary (SYSIN) input with the exception that further INCLUDE directives within INCLUDE'd source are not allowed. The same source margins are applied to the INCLUDE'd source as are applied to the primary input. In addition, the compiler prints a line in the primary source listing indicating the catenation sequence number of the DD card on which the member was found and the RVL field from the PDS directory entry for the member. The RVL field is the first 2 bytes of user data after any TTRN's.

The individual members which are INCLUDE'd may be in either compressed or uncompressed format, independent of whether the primary input was compressed. The form of each INCLUDE'd member is determined by the compiler from the first record read.

Partitioned datasets may be catonated together in the JCL to form the INCLUDE DD sequence, but such datasets must have identical DCB attributes.

# 2.3 ACCESS Rights Implementation

The HAL/S language allows managerial restrictions to be placed upon the usage of user-defined variables and external routines. The existence of such a restriction is indicated by the use of the ACCESS attribute as described in the HAL/S Language Specification. The manner in which the restrictions are enforced in the HAL/S-FC compiler system is described below.

Any variable in a COMPOOL template or any external routine to which the ACCESS attribute has been applied is considered to be restricted for the compilation unit which is being compiled. The restriction is slightly different for variables than for blocks:

- a) Variables with the ACCESS attribute may not have their values changed.
- b) Block names may not be used at all.

These restrictions may be selectively overridden for individual variable and block names. The selection of which ACCESS controlled names are to be made available to the unit being compiled is performed by processing an external dataset. The external dataset is known as the Program Access File (PAF). The PAF must have partitioned organization and is specified by the following JCL:

//HAL.ACCESS DD DSN=<PAF name>, <other parameters>

where the <PAF name> is the dataset name of the PAF without any member specification.

Each member of the PAF contains the information about ACCESS controlled names which are to be made available to one unit of compilation. The member name is defined by a Program Identification Name (PIN). The PIN is specified to the HAL/S-FC compiler by using the PROGRAM compiler directive in the primary input stream:

col 1

D PROGRAM ID = <id>

The <id> field of the directive is a 1 to 8 character identifying name which is used to select the member of the PAF to be processed for the current compilation's ACCESS information. The appearance of the PROGRAM directive in the compiler's input stream causes immediate processing of the PAF member specified.

The format of an individual PAF member is described below.

- a) Column 1 of each record is ignored except when column 1 contains the character "C", in which case the entire record is ignored.
- b) The portion of each record which is processed is the same portion which is processed in the primary compiler input (SYSIN).
- c) COMPOOL elements which are to be made available to the compilation are specified as:

<COMPOOL-name> (<var-name>, <var-name>, ... <var-name>)

or

<COMPOOL-name> (\$ALL)

The first format specifies access to individual variables within the named COMPOOL. The second format specifies access to all variables within the named COMPOOL.

- d) Access to external block names is specified as: \$BLOCK(<ext-name>, <ext-name>)
- e) Blanks are allowed anywhere in the record except that names may not be broken by a blank.
- f) Either of the constructions (c) or (d), above, may span more than one record.
- g) The name of the particular COMPOOL in the form (c) above may appear more than once; i.e. the variables in a particular COMPOOL do not have to be specified at one time. Similarly, the form \$BLOCK may appear more than once.

Some validity checking is performed by the compiler while processing the PAF member. Warnings are issued for the following conditions:

- 1) A syntax error on a PAF record the bad record is printed.
- 2) Names mentioned in the PAF are not defined.
- 3) Elements of \$BLOCK in the PAF are not defined.

- Requests for names which are not ACCESS 4) protected.
- Variables found, but not within the COMPOOL 5) specified.
- Names used in the context of a COMPOOL-name which are not COMPOOLs.

If, at the time the PROGRAM directive is encountered, there have been no ACCESS-controlled variables declared, the PAF is not opened. If a user does not require access to any, the PROGRAM directive and associated PAF members may be omitted.

#### Compiler Directive Parsing 2.4

When an input record is found which contains a "D" in column one, Phase I scans the remainder of the card for a valid compiler directive. A list of legal compiler directives and their function is listed in Appendix B.

Directive processing is done independently of HAL/S source language parsing, i.e. words used on Directive cards are not necessary HAL/S language keywords. Similarly, HAL/S language keywords are not recognized as such on Directive cards.

#### Template Checking and Generation 2.5

Phase I assumes the task of source template verification and generation. Every compilation unit in the HAL/S-FC system has a source template. When the block header for a unit of compiler is encountered, Phase I begins to construct the source template for that unit as follows.

The member name for the template being created is determined. This is done by taking the "characteristic name" for the unit and preceeding it by the characters '00'. The characteristic name for any unit is created by taking the block name, removing any underscore characters, and then padding or truncating the result to 6 characters. An attempt is made to locate a member of this name on either the INCLUDE or OUTPUT6 DD cards. such a member is found, the contents of the member are compared with an internal, temporary template created as the compilation proceeds. If the existing template and the internal one

agree, a template update is not required, and the existing template remains intact. If the templates do not agree, the internal template is written to the OUTPUT6 DD card and STOW'ed with the current member name. If the initial search for an existing template fails, the generated template is automatically written and STOW'ed on the OUTPUT6 DD card. The PDS directory entry for a template member is created with two bytes of user data. The two bytes are initialized to X'FOFO'.

Phase I also sets appropriate bits in a field which is passed back to the caller of the compiler as the high order byte of register 15. The definitions of these bit settings is defined in the HAL/SDL ICD.

Generation of the internal template is performed during syntax analysis on a token by token basis. As statements are encountered which are required in the template, the tokens from the statements are added to an internal buffer. When a new token will no longer fit in the buffer, the buffer is written and clearned for continuation. Thus, the templates take the form of strings of HAL/S tokens separated by one block. The template statements are continued from one line to the next without regard for statement boundaries, thus producing the template in the most compact form possible.

For the comparison of existing templates with new, generated templates, the generated records are compared character for character with the existing records. Any mismatch is considered to indicate a change in the template.

Templates are never generated using the compressed source format mentioned in Section 2.1. The generated templates conform to the source margins in effect for the compilation (e.g. for an SDL mode compilation, templates are created with source in position 2 through 72 of the records. When template records are written to the OUTPUT6 DD card, the records are padded with blanks or truncated as necessary to conform to the LRECL specification for that DD card.

When a template has been found to have changed, the compiler updates a "Version number" associated with the template. For an existing template, the version number is found on a VERSION compiler directive card at the end of the existing template member. If a new template is needed, the version number is incremented by one and placed on a new VERSION directive card at the end of the generated template. The version number is limited to the range 1 to 255. Upon reaching 255, the next incrementation causes the number to begin again at 1. When no existing template can be located, the version is set to 1.

When templates produced by the compiler are referenced in subsequent compilations by use of an INCLUDE for the template, the version numbers from the referenced templates are emitted into the produced object code on special SYM records which indicate the versions of all external references. In addition, the emitted object code for any compilation unit contains a SYM record indicating the version number of the template created for that compilation unit. This information permits the checking, if desired, of proper integration of separately compiled units by providing information necessary for cross-checking of inter-module references.

### 2.6 Listing Generation

### 2.6.1 Primary Formatted Listing

The central printed output of the compiler is the primary source listing. This listing is designed to document the actions taken by the compiler during its generation of an executable form of the user's program. The listing reproduces the user's source program in an indented, annotated format. Additional information, such as block summaries and symbol table listings, are also part of the primary source listing.

The formatting of the primary source listing leads to the documentation of the users program in two ways: 1) variable annotation, and 2) logical indenting.

- 1) Variable annotation Each user-defined data symbol, when printed on the primary source listing, receives "marks" appropriate to the type and organization of the symbol. This annotation is that which is defined by the HAL/S Language Specification.
- 2) Logical indenting Each statement printed on the primary source listing is formatted and indented to show internal statement structure, and to show the statements' hierarchical and nesting relationships to other statements in the compilation.

When operating in the SDL, additional information is provided on the primary source listing. The Record Sequence Number, Record Revision Indicator, and Change Authorization fields (see Section 2.1) are printed on the primary source listing next to the statements to which they apply. Additional details of the specific operations performed during SDL operation may be found in the HAL/SDL Interface Control Document.

#### 2.6.2 Error Messages

When compilation errors are detected by Phase I, an error message is printed in the primary listing at the point of detection. All error messages have an identifying code associating with them.

The code is assigned to messages according to a general system which groups errors according to a class and a subclass. Multiple errors within a class/subclass combination are assigned unique numbers within the group. Thus, every possible error in the HAL/S-FC compiler system has a unique identifying code. Appendix C lists the error classification scheme.

The text of all error messages is maintained on a direct access dataset. The compiler retrieves error message text as needed from this dataset. During compilation, the ERROR DD card defines the error message dataset. This file has partitioned organization and contains one member for each error message. The member names are identical to the identifying code assigned to the errors.

The record format of the error library is FB and the logical record length is 80 bytes. The first record of each member defines the severity of that error. The severity is a single EBCDIC number in position one of the first record. The severities and their effects are:

- 0 = warning (compilation proceeds)
- 1 = error (further compilation attempted)
- 2 = severe error (Phase I syntax check proceeds; code generation prevented)
- 3 = abortive error (compilation halts immediately)

Within the text of an error message, locations into which specific descriptive information may be placed are denoted by the appearance of two question marks (??). For errors which have this feature, the compiler supplies additional description text (such as the name of an identifier) to make the printed error message as specific and informative as possible.

#### 2.6.3 Block Summaries

The HAL/S-FC compiler provides additional information on the primary listing at the close of HAL/S code blocks. The blocks for which summaries are given are PROGRAM, TASK, FUNCTION and UPDATE.

Information contained in block summaries consists of lists of labels or variable names used in various contexts within the block. The title "BLOCK SUMMARY" begins the list. For all potentially summarized contexts within the block, a descriptive heading is printed followed by the list of names involved. A "\*" next to any name in the block summary indicates that the name appears in a context which changes its value. The headings are listed below.

PROGRAMS AND TASKS SCHEDULED

PROGRAMS AND TASKS TERMINATED

PROGRAMS AND TASKS CANCELLED

EVENTS SIGNALLED, SET, OR RESET

EVENT VARIABLES USED

PROGRAM OR TASK EVENTS USED

PRIORITIES UPDATED

EXTERNAL PROCEDURES CALLED

EXTERNAL FUNCTIONS INVOKED

OUTER PROCEDURES CALLED

OUTER FUNCTIONS INVOKED

ERRORS SENT

COMPOOL VARIABLES USED

COMPOOL STRUCTURE TEMPLATES USED

COMPOOL REPLACE DEFINITIONS USED

OUTER VARIABLES USED

OUTER REPLACE DEFINITIONS USED

OUTER STRUCTURE TEMPLATES USED

### 2.6.4 Compilation Layout Summary

Immediately preceding the Symbol Table printout at the CLOSE of the HAL/S program, there is a compilation layout map, indicating the way in which PROGRAMS, TASKS, PROCEDURES, FUNCTIONS, and UPDATE blocks were defined. The indent level in this printout indicates the nesting level definition of the block shown. This serves to give a quick overview of the compilation structure.

### 2.6.5 Symbol & Cross Reference Table Listing

The symbol and cross reference table printed at the end of a HAL/S compilation listing provides a detailed accounting of all programmer-defined symbols. The table listing is organized into two parts: a structure template listing and an alphabetized total listing.

Any structure templates defined in the compilation appear first in the symbol and cross reference table. The template names appear in alphabetical order. The body of each template (i.e. the levels defined under the template name) is listed under the template name in the order of definition. This ordering provides a quick reference to the organization of the structure template.

Following any listing of the templates, an alphabetized listing of all programmer-defined symbols is printed. Symbols previosuly listed as element of a structure template are included in this list. However, the list is completely alphabetized and template organization is not shown. When a particular symbol is independently defined in more than one name scope, the symbol is multiply listed in order of definition.

#### 2.6.6 Built-in Function Cross Reference

Phase I also produces a listing of any HAL/S built-in functions used in a compilation. The printout shows the statement numbers at which the references to the built-in functions occurred.

### 2.6.7 Replace Macro Text

If any HAL/S REPLACE statements were used in the compilation, the text of the macro is printed in the symbol table listing in the attributes and cross reference area.

### 2.6.8 Unformatted Source Listing

Under control of the "LISTING2" compiler option,
Phase I will optionally produce, on the file defined by
the LISTING2 DD card, a listing of the input (both SYSIN
and INCLUDE) source records as read by the compiler. No
special annotation, formatting, or indenting is performed.
In the case of input in the SDL compressed format, the
LISTING2 option produces the records in their uncompressed
format.

### 2.7 Symbol Table Generation

Phase I is responsible for initial creation of the compiler's internal symbol table. The symbol table consists of a group of arrays which describe all of the properties of declared variables and labels. The capacity of the symbol table is under user control by means of the SYTSIZE compiler option. This table, as created by Phase I, is located in an area common to all compiler phases. Thus, Phase II inherits the initialized table from Phase I.

Design of the HAL/S-FC compiler includes, as a basic concept, the use of a Phase I and Phase I/Phase II interface identical to that of the HAL/S-360 compiler. Thus, the description of the internal symbol table to be found in the HAL/S-360 Compiler System Specification, Appendix B.2 is sufficient to define the HAL/S-FC table.

### 2.8 Statement Table Generation

The statement table passes information about executable statements from Phase I of the compiler to Phase III. This information allows Phase III to include statement type and target variable information in the Simulation Data Files.

Due to the use of a common Phase I in the HAL/S-360 and HAL/S-FC compiler systems, the Statement Table description in the HAL/S-360 Specification document is sufficient to describe the HAL/S-FC table. (See Appendix B.3 of that document).

The basic table description includes reference to an "extension" field in which statement memory addresses and/or SRN data is stored. Use of this field is activated by use of certain compiler options:

SRN data is included in the Statement Table if either of the SRN or SDL compiler options are used.

Beginning and ending addresses for individual HAL/S statements are included in the Statement Table when the ADDRS compiler option is used.

The Statement Table is produced on the file specified by the FILE6 DD card. No Statement Table data is communicated via in-memory tables.

### 2.9 Literal Table Generation

The format of the HAL/S-FC literal table is identical to that used by the HAL/S-360 compiler as described in Appendix B.1 of the HAL/S-360 Compiler System Specification.

The size of the area in which character literal data is stored is under user control via the LITSTRINGS compiler option. This character literal area is communicated to subsequent phases of the compiler through common memory locations.

The portion of the literal table which contains arithmetic literals, bit literals, and pointers to character literals is passed to later phases via the dataset defined by the FILE2 DD card.

### 2.10 HALMAT Creation

HALMAT is the intermediate code medium by which the structure of the compiled HAL/S program is passed to Phase II for code generation. The HAL/S-FC compiler uses the same Phase I as the HAL/S-360 compiler. Therefore, the HALMAT produced by Phase I for either system is the same. A description of HALMAT as used by these compilers can be found in Appendix A of the HAL/S-360 Compiler System Specification.

HALMAT is passed to Phase II through use of auxiliary storage as defined by the FILE1 DD card.

### 2.11 The Optimizer

The HALMAT produced by Phase I is a direct representation of the HAL/S program being compiled. A separate phase of the compiler exists between Phases I and II which examines and manipulates the HALMAT in order to produce an optimized HALMAT representation. This phase, known as Phase 1.5, is conceptually a part of Phase I. Its operation is transparent to the user as it produces no standard printouts.

The Optimizer performs the following functions:

- Common subexpression elimination
- Additional literal folding
- Replacement of unneeded divisions by multiplications
- Supression of unnecessary matrix transpose operations

- Indication of procedures which cannot be leaf procedures (as an aid to Phase II).

These operations are carried out by modifying the HALMAT, literal table, and symbol table.

While the Optimizer is a separate phase, it is conceptually a part of Phase I and is described in the HAL/S -360 Compiler System Specification.

### 3.0 PHASE II - CODE GENERATION

The code generation phase of the HAL/S-FC compiler has the primary function of producing machine language instructions for the AP-101. Phase II also performs other tasks which are also the subject of this chapter.

This section deals with the following Phase II functions:

- Code Generation
- Naming Conventions
- Printed Data
- Symbol Table Augmentation
- Statement Table Augmentation

### 3.1 Code Generation

### 3.1.1 Bases and Conventions

Phase II produces AP-101 machine language instructions which perform the operations indicated by each line of HALMAT received from the syntax and semantic analysis phase. This section describes in detail the ground rules which the code generation phase follows in producing object code. The following terms will be used throughout the ensuing text:

- R A general accumulator (integer or scalar);
- X An indexing register (for subscripting);
- B A base register containing a base address used to compute the effective address of a variable, constant, temporary, or program label.
- OFFSET The constant term which, when subtracted from the actual data address of a variable, yields the address of the 0'th item of the aggregate data collection (note that all HAL subscripts start counting from 1). This is 0 when the variable is a single item.
- VAR The address of a declared non-parameter HAL variable. For addressing purposes, it is actually the base address of the actual data minus the OFFSET. Single valued integer, scalar, or bit input parameters also will use this form.
- PAR The address of a formal parameter passed "by reference". This includes any assigned parameters, plus any input parameters which are not simple integer or scalar variables. Note that PAR actually contains an address.

1

- DELTA The constant indexing term in a subscript calculation. This term may also reflect the displacement of a structure terminal within a structure template.
- OP Any AP-101 machine instruction.

Note - When VAR or PAR appears in machine instruction constructions, it represents the displacement difference between the data address and the base address contained in the base register B.

- 3.1.1.1 Register Usage. The following register assignments are used by the code generator:
  - F0-F5 Used for floating point accumulators and parameters.
  - F6-F7 Used for floating point accumulators only.
    - O Stack register. This register points to the caller's register save area in the run time stack. In addition, all formal parameters, temporaries, and AUTOMATIC variables in REENTRANT procedures are based on this register.
    - Global data addressing register. This register is used to address all of the declared variables and literals within a compilation unit.
    - Work addressing register. This register is used to pass address parameters, dereference NAME variables, and set up any other dynamic addressing.
    - Local addressing register. This register is used in SRS instructions only to address a certain subset of the local data in a block.
    - Linkage register. This register records the return address for all subroutine linkages.

      It may also be used for an integer accumulator.

5-7 Used for integer accumulators, index registers, and parameter passage where applicable.

3.1.1.2 Storage Allocation. The HAL/S-FC compiler arranges data in memory such that the least number of base registers need be dedicated in addressing.

Data is grouped into two major categories: single value (constant offset=0) and aggregate (array, vector-matrix, structure with copies). Within in each group, data is ordered such that data requiring the same boundary alignment is adjacent, minimizing the storage lost due to hardware alignment requirements. Within the aggregate group, ordering is further carried on such that multi-dimensional arrays (with larger offsets) come after single dimensional arrays. These above orderings are carried on independently for: 1) program data, and 2) each COMPOOL block contained in the compilation unit. Note that program data includes all variables within the compilation unit including those defined in procedures, functions, or any other block.

Structure templates are internally ordered such that the minimum boundary alignment within any node level is required. Template matching requirements guarantee that templates exhibiting identical properties will be identically reordered.

After all groupings are complete, storage assignments are made, with the required base-displacement combinations being generated to properly access the data. Note that the storage addresses assigned refer to the actual data beginning, but the base-displacement address includes the negative OFFSET value.

Note that all formal parameters and all AUTOMATIC variables in a REENTRANT PROCEDURE or FUNCTION are based off the stack register (0).

For arrays, the offset is computed as follows for the number of array dimensions: ( $N_i$  is the i<sup>th</sup> array dimension).

4	ŧ	D	i	m	93. 33.4							0	££	s	e	=					
				!									,								
		C	)										١	,							
		1	• .										-]	L							
		2	2			٠.				ď,	(-	1	1	12	) ·	-1					
		-	ł.											-	ь.	) -	) N	1,	) .	<b>-</b> ]	L
		-	•								•	•			Z		*.	ು			

The array OFFSET is then multiplied by the total width of the data type specified. For integers, scalars, bits, and characters, this is the width in halfwords to contain one item of data. For vector and matrix types, this is the width in halfwords for one item times the total number of items in the vector or matrix.

For structures, the OFFSET is 0 if the structure has no copies. If the structure has copies, the offset is -W, where W is the aligned width of one copy of the structure template.

#### Example:

DECLARE	A SCALAR,
	B INTEGER,
	C CHARACTER (7),
	D ARRAY (5) DOUBLE;
DECLARE	E ARRAY (5),
	F ARRAY (3,3) VECTOR,
	G MATRIX;
DECLARE	H DOUBLE,
	I ARRAY (5,5) INTEGER;

Alignment	NAME	Location	Base	Displacement	(In Decimal) Offset
Halfword	В	00000	1	0000	0
Halfword	C	00001	1	0001	
Fullword	Α	00006	1	0006	0
Doubleword	H	80000	1	0008	
Halfword	I	0000C	1	0006	-6
Fullword	E	00026	1	0024	-2
Fullword	G	00030	1	002E	<b>-2</b>
Fullword	F	00042	1	0028	<b>-26</b>
Doubleword	D	00078	1	0074	-4

3.1.1.3 Addressing Concepts. This section describes the general addressing rules for data. To the extent possible, data can be directly addressed via some combination of base register and bit displacement (eleven bits for indexed addressing). This is not possible whenever the data item is a formal parameter other than a simple integer or scalar, or any formal parameter scoped in from an outer to an inner procedure. The skeletal forms given in Section 3.2.2 assume the most commonly used addressing forms. The rules described here should be superimposed upon these skeletal forms to interpret all possible combinations of operations between operands.

### Simple Addressing Forms

Simple Variable

OP R, VAR(B)

Simple Aggregate Component

(array or vector-matrix)

OP R, VAR+DELTA(X,B)

Simple Integer-Scalar formal parameter

OP R, VAR(0)

Simple Aggregate formal parameter

L B, PAR(0)

OP R, DELTA(X,B)

NAME Variable in de-reference context

LH B, VAR(B)

OP R, DELTA(X,B)

NAME Variable in de-reference context (ASSIGN formal parameter)

L B, VAR(B)

LH B, 0(B)

OP R, DELTA(X,B)

REMOTE Variable

OP@# R, ZCON(X,1)

ZCON DC Z(0, VAR, 0)

### Scoped Formal Parameter Addressing Forms

For scoped formal parameters, generation of an effective address requires a loop to trace references back through multiple save areas in the run time stack. In both of the scoped formal parameter sequences below, the offset of 0 in the load instruction at the head of the loop represents the fixed location of the next higher level's register copy. The loop terminates when the nest level of the parameter in question is equal to the nest level of the current save area being referenced.

```
Scoped Integer-Scalar formal parameter:
```

LHI 4, <scope number of parameter>

LR 2, 0

LOOP L 2, 2(2)

CH@ 4, 9(2)

BNE LOOP

OP R, VAR(2)

Scoped Aggregate or NAME formal parameter:

LHI 4, <scope number of parameter>

LR 2, 0

LOOP L 2, 2(2)

CH@ 4, 9(2)

BNE LOOP

LH 2, PAR(2)

OP R, DELTA(X,2)

Scoped NAME ASSIGN formal parameter:

LHI 4, <scope number of parameter>

LR 2, 0

LOOP L 2, 2(2)

CH@ 4, 9(2)

BNE LOOP

LH 2, PAR(2)

LH 2, 0(2)

OP R, DELTA(X,B)

### Address passage addressing forms

For parameter passing to PROCEDURES, FUNCTIONS, and library routines, it is often necessary to pass address pointers instead of data. The following sequences could be used anywhere the instruction LA appears in the generated code sequence (including NAME assignments).

Unsubscripted variable:

LA R, VAR(B)

Subscripted variable:

SLL X, <index alignment>

LA R, VAR(X,B)

Unsubscripted REMOTE variable:

L R, ZCON(1)\*

Subscripted REMOTE variable:

SLL X, <index alignment> or SLL R, <index alignment>

L R, ZCON(1)\* A R, ZCON(1)\*

AR R, X

Unsubscripted variable to REMOTE library:

LA R, VAR(B)

IAL R, X'0400'

Subscripted variable to REMOTE library:

SLL X, <index alignment>

LA R, VAR(X,B)

IAL R, X'0400'

<sup>\*</sup> ZCON DC Z(0, VAR, 0).

### Indexing:

The computation for all indexing is done as follows. All constant index terms are factored out from the variable terms. The variable terms are computed according to the natural sequence of unwinding aggregate data. The constant terms are similarly computed to form a DELTA. The variable subscript in register X is shifted according to the halfword width of the data being indexed, except for those instructions which perform automatic index alignment. The DELTA is similarly shifted at compile time. If  $0 \le \text{DELTA} \le 2048$ , it is used in the variable displacement. Otherwise, it is added to X if X is non-zero, or loaded into a newly created X if X is zero (i.e. the subscript contains no variable terms).

3.1.1.4 Condition Codes. The following table lists the allowable relational operations and the resultant condition code - referred to as COND throughout the remainder of this section. Note that the AP-101 conditional branch instructions branch on the "not true" condition.

<u> <c< u=""></c<></u>	<u>)P&gt;</u>	COND
_		3
		4
<		5
>		6
<b>- &lt;</b>	or > =	2
¬ >	or < =	1

3.1.1.5 ZCONS and the Calling Mechanisms. Throughout the descriptions of generated code of Section 3.1.2, branches to other CSECTs (comsub or library) are generally indicated as:

#### ACALL <routine name>

The actual implementation of this linkage is to go not directly to the named routine, but instead to branch indirectly through a long address constant (ZCON) located in sector 0 of the machine.

When the target of the branch is a compiler-generated CSECT (a COMSUB), the ZCON referenced will be one created during compilation of the COMSUB. The long indirect address will be in a CSECT named #Znnnnnn (see Section 3.2) which will in turn refer to the real code CSECT.

When the target of the branch is a library routine, the ZCON referenced will be one provided with the library. Its name will be #Qnnnnnn and it will in turn refer to the proper library code CSECT. Certain library routines, for reasons of execution speed, are referenced directly by compiler-emitted code without going through a ZCON. These routines are designated in the BANKO column of the library documentation. This direct addressing requires that these routines reside either in sector zero or in the same sector as the compiler code which references them.

The use of ACALL in the descriptions implies an external call. In actuality, the instruction generated may be either:

SCAL 0, <routine name>

or

BAL 4, <routine name>

depending on whether the library routine has been designated as PROCEDURE or INTRINSIC type.

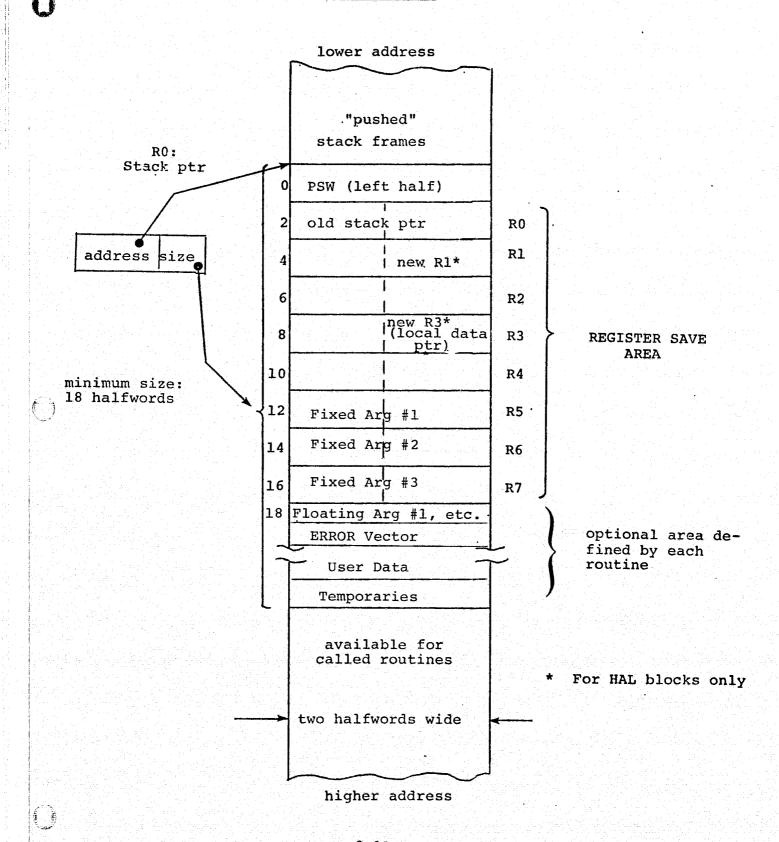
Some of the parameter setups show the use of P1, P2, and P3 for parameter registers. The following table shows the actual register values for P1, P2, and P3 depending upon the nature of the library routine (see library documentation for specific details).

	Pl	P2	P3
Intrinsics	1	2	3
Procedure -		Ja 7	J
Pl used	2	4	7
Pl not used	X	2	4

3.1.1.6 The Runtime Stack. The HAL/S-FC compiler system employs a runtime stack mechanism as an integral part of its operation. The stack mechanism is used to provide subroutine linkage areas, temporary work areas, error environments, and to provide reentrancy of code blocks when needed. The actual memory used as a stack space for a given HAL/S process is provided by the flight computer operating system The determination of the size required for a particular stack is made by the flight computer support software linkage editor. The linkage editor determines stack size (and upon special request will create a stack CSECT) from information provided on SYM cards in the modules being link edited. The HAL/S-FC compiler emits the SYM cards as part of its object modules. The runtime library uses a system of macros to generate the properly named DSECT's and SYM entries for stack size computation.

The details of formats and requirements relating to stack generation can be found in the HAL/SDL Interface Control Document. That document also contains the detailed description of the "stack frame", that portion of a total stack which is used by an individual subroutine when that subroutine has been invoked. The description of the basic stack frame is reproduced here for reference.

The active stack frame is pointed to by the pointer in register R0. The back link to the previous stack frame is established when a new level is entered. A pointer, NEW R3, is established for any block with a local data area. If a local data area is not present, e.g. in the case of a HAL/S-FC library routine, NEW R3 is set to zero. See Section 3.1.1.7 for a definition of the local data area.



3.1.1.7 Local Block Data Areas. During execution of a HAL/S-FC program, certain machine registers have dedicated uses as described in Section 3.1.1.1. In particular, register R3 is a local addressing register which points to the local Block Data Area for the block in execution. These R3 values are saved on the runtime stack as indicated in Section 3.1.1.6. The format of a local Block Data Area is the subject of this section. The HAL/SDL ICD contains the controlling definitions of these areas.

Block Data Areas are created by the compiler and are part of the #Dnnnnnn CSECT generated for a compilation unit. A Block Data Area may exist for any Program, Procedure, Function, Update Block, or Task. The compiler-emitted code for block entry (as defined in Section 3.1.9) loads R3 with the address of the Block Data Area for the block being entered. The format of such an area is shown in the following diagram.

•	• -	1 -1	
н	10	ICI	S

BL 1	Block	ID		2	
2	ХU	ONERRS	ERRDISP	2	
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	T Y P	UNUSED	RESERVE SVC #	2	
4	UNUS	ED	RELEASE SVC #	2	only required if XU = 1
<b>5</b>	LOCK	ID		2	

#### Field

#### Definition

1. Block ID

A 16 bit field uniquely identifying the HAL block. The first 9 bits are a "compilation number" supplied by the user via the COMPUNIT compiler option. The last 7 bits are a block count generated internally for each new block within a compilation unit.

Field

### Definition

2. XU

EXCLUSIVE/UPDATE flag. (1 bit). Set to one if block is either an UPDATE block or has the EXCLUSIVE attribute.

ONERRS

(6 bits). The number of discrete errors for which an ON ERROR statement exists in the block.

ERRDISP

(9 bits). The displacement in half words from the stack frame pointer register (R0) to the error vector

3. TYP

(1 bit). Set to zero for EXCLUSIVE procedure or function. If an UPDATE block, set to one if shared data variables are read only. Set to zero if shared data variables are to be written.

Reserve SVC# (8 bits). SVC number for the reserve SVC:

15 for a code block

16 for a data area.

- 4. Release SVC# (8 bits). SVC number for release SVC:
  17 for a code block
  18 for a data area.
- 5. Lock ID (15 bits). An indicator of which code block or data areas are being used. For a code block this is the address of the EXCLUSIVE DATA CSECT of the procedure/function. For a data area this is a bit pattern indicating which data areas (by lock groups) are involved. If the "master lock" was specified, the bit pattern will be all ones.

41

3.1.1.8 Parameter Passing Conventions for User-written Routines. To the extent possible, HAL/S parameters are passed via registers. Scalar parameters are passed in floating point registers. All others are passed in general registers. The following rules describe how the registers are designated, and what they contain for each type of parameter.

General purpose registers 5-7 and floating point registers 0, 2, 4 are available for parameter passing. If the supply of registers is exhausted before the parameter list, the balance of the parameters are passed in memory locations. All parameters are located via the stack register (0).

Allocation of general and floating registers is carried on in parallel. If no scalar parameters exist, no floating point registers will contain parameters.

General purpose registers 5 through 7 are automatically contained in the stack beginning at displacement 1210. Floating point registers are not automatically saved, and it is the responsibility of the called program to do so. Storage locations are reserved in the stack for this purpose as described below. Parameters which cannot be passed in registers are automatically stored in the called procedure's stack by the caller. The allocation of these stack locations is identical to the allocation for floating point values. Note that, unlike ordinary HAL/S variable allocation, parameter allocation is not subject to reordering to minimize alignment conflicts.

The first available stack location is at  $18_{10}$  off the stack register. All parameters are assigned storage in order starting at this point (the exception being parameters contained in general registers 5 through 7, which are allocated space in the register save area as described above). Any necessary alignment is performed as needed.

Arguments are either input type or ASSIGN type. (Input types are those whose values will not be changed by the called routine.) The actual information which is passed for a particular argument is dependent upon the following factors:

- whether the argument is input or ASSIGN;
- whether the HAL/S data type of the argument is an aggregate (i.e. more than one element, as in a matrix);
- whether the argument has any arrayness or structure copies to be passed; and
- whether any arrayness or structure copies are defined via an ARRAY(\*) or -STRUCTURE(\*) specification.

The following table and list show the information which is passed for an argument with particular attributes.

Data Type Argument Type		Scalar	Bit	Character(*)	Vector	Matrix	Structure
Input (no arrayness or copies)	1	2	3	<b>4</b> .	5	6	7
ASSIGN (no arrayness or copies)	8	8	8	4	5	6	7
Input or Assign (with arrayness or copies)	9*	9*	9*	10*	9*	9*	11*

<u>Key</u>	Information Passed
	A halfword or fullword of data.
	A single or double precision floating point value.
	Up to 32 bits of data (halfword or fullword depending upon declared size).
	Address of the max-size byte of the character string.
5	Address of the 0 <sup>th</sup> item in the VECTOR (i.e. 1 item width ahead of the actual vector).
6	Address of the 0 <sup>th</sup> item as if the MATRIX were a VECTOR of length m x n.

### REPRODUCIBILITY OF THE ORIGINAL PAGE IS 100 TO

<u>Key</u>	Information Passed
7	Address of the first location in the structure as defined by its template. (Note that item position within a template is subject to compiler reordering unless RIGID is used).
8	Address of the data item.
	Address of the 0 <sup>th</sup> item of the array.
10	Two items are passed. The first is the address of the 0 <sup>th</sup> array item. The second is the number of halfwords of memory occupied by one character string element (including the halfword containing the max and current size bytes).
11	The address of the first data in the 0 <sup>th</sup> copy.

\* If the parameter is declared as ARRAY(\*) or -STRUCTURE(\*), an additional parameter word is passed containing the value of the unspecified dimension.

For all cases where auxiliary values are allocated for a single parameter (i.e. CHARACTER(\*) ARRAY or ARRAY(\*)), the parameters (up to 3) must be contiguous. Thus, if more pointers are required than registers are available, then the whole parameter sequence will be pushed into the stack.

#### Example:

P: PROCEDURE (X, Y, I, J, K, Z, C, L);

DECLARE SCALAR, X, Y, Z DOUBLE;
DECLARE INTEGER, I, J ARRAY(\*), K, L;
DECLARE CHARACTER(\*) ARRAY(\*), C;

Upon entry to this procedure, the stack and registers are as follows:

	$\leftarrow$ 1 word $\rightarrow$					
R1+12 <sub>10</sub>	I unused	a]	.so	in	R5	
+14	address of 0 <sup>th</sup> array element of J	a]	.so	in	R6	
+16	size of array J unused	al	so	in	R7	
+18	<b>X</b>	al	.so	in	F0	
+20	Y	al	.so	in	F2	
+22	K unused					
+24	1 <sup>st</sup> word of Z	)		4	F4,	13 C
+26	2 <sup>nd</sup> word of Z	di	SO	ти.	Г4,	ГЭ
+28	address of 0 <sup>th</sup> array element of C					
+30	# HW occupied by one element of C					
+32	Size of array C					
+34	L unused					

#### 3.1.2 Integer and Scalar Operations

#### Nomenclature

The register R is any of the available set of accumulators. The terms I,  $I_2$ , S, and  $S_2$  refer to the single and double precision versions of Integer and Scalar values respectively. It is assumed that any implicit precision or type conversions have been accomplished prior to generating the code sequences shown below.

3.1.2.1 Arithmetic Operators. Integer and scalar arithmetic operators generally employ two operands, denoted as X and Y. X is assumed to be loaded into register  $R_{\rm X}$  unless otherwise noted. If Y is also in a register, it is represented by the form  $R_{\rm Y}$ .

Operation	Туре	Co	<u>de</u>	Alter	nate	Code
X + Y:	1	АН	R <sub>x</sub> , Y	AHI	R <sub>x</sub> ,	Y*
	12	A	R <sub>x</sub> , Y	AR	R <sub>x</sub> ,	Ry
	<b>. S</b>	AE	R <sub>x</sub> , Y	AER	R <sub>x</sub> ,	Ry
	<b>s</b> <sub>2</sub>	AED	R <sub>x</sub> , Y	AEDR	R <sub>x</sub> ,	R Y
x - Y:	Similar to operator is place of AH	used	<ul> <li>(For example)</li> </ul>	umple,		
(Multiply) X Y:	I	мн	R <sub>x</sub> , Y	МІН	R <sub>x</sub> ,	<b>Y*</b>
		SLL	R <sub>x</sub> , 15			
	12	M	R <sub>X</sub> , Y	MR	R <sub>x</sub> ,	Ry
		SRDA	R <sub>x+1</sub> , 1			
	S	ME	R <sub>x</sub> , Y	MER	R <sub>x</sub> ,	Ry
	<b>s</b> <sub>2</sub>	MED	R <sub>x</sub> , Y	MEDR	R <sub>x</sub> ,	R <sub>y</sub>

Note that the shift operations used in the integer multiplications are required to correctly normalize the result in the proper registers.

Certain constant multipliers are optimized to avoid using actual multiply instructions. They are described below.

<sup>\*</sup> Used if Y is a literal.

Operation	Type	Code	Alternate Code
	1 2 <sup>n</sup>	SLL R <sub>1</sub> , n , n>1	
		$AR \qquad R_{I}, R_{I}, n=1$	
	$I_2^2$	SLL $R_{I}$ , $n$ , $n>1$	
		$AR \qquad R_{I}, R_{I}, n=1$	
	<b>x</b> 1	no code for any type	
	S 2	AER R <sub>s</sub> , R <sub>s</sub>	
	s <sub>2</sub> 2	AEDR R <sub>s</sub> , R <sub>s</sub>	
<b>X/Y:</b>	S	SER R <sub>x</sub> +1, R <sub>x</sub> +1	SER R <sub>x</sub> +1, R <sub>x</sub> +1
		DE R <sub>x</sub> , Y	DER R <sub>x</sub> , R <sub>y</sub>
	s <sub>2</sub>	DED R <sub>x</sub> , Y	DEDR R <sub>x</sub> , R <sub>y</sub>
X**Y:	patterns which w	onentiation is performed is shown for I and S are in ill be generated for I2 a ious differences.	deutical to those
		LH 5, X ACALL HTOE* LH 6, Y	
	S**I	LH 6, Y (see note) LE 0, X	Argument Setup
	S**S	LE 2, Y LE 0, X	
		ACALL αPWRβ	} Actual Call

where  $\underline{\alpha}$  and  $\underline{\beta}$  represent the types of operands X and Y respectively:

Type of X	α	Type of Y β
single precision integer double precision integer single precision scalar double precision scalar single precision integer double precision integer	} E     D     H*     I*	single precision integer H double precision integer I single precision scalar E double precision scalar D

Return is in F0 for  $\alpha$  of E or D; in R5 for  $\alpha$  of H or I.

\* If Y operand is a positive integer literal, the HTOE conversion is eliminated and the PWR routine invoked is  $\alpha PWRH$  or  $\alpha PWRI$ .

peration	Type	Code Alternate Code	<u>e</u>
Note:	powers fr repeated powers al	pressions raised to integer literal om 1 to 16 are performed in-line via multiplication, using the binary gorithm. The following examples should be shown that the method.	
X**1:		No code generated.	
X**2:	<b>S</b>	MER R <sub>x</sub> , R <sub>x</sub>	
X**3:	s	LER $R_{T}$ , $R_{x}$	
		MER R <sub>X</sub> , R <sub>X</sub>	
		MER $R_{T}$ , $R_{x}$	
		(result in $R_{\overline{\mathbf{T}}}$ )	
X**6:	S	MER $R_{x}$ , $R_{x}$	
		LER R <sub>T</sub> , R <sub>X</sub>	
		MER R <sub>X</sub> , R <sub>X</sub>	
		MER $R_{T}$ , $T_{x}$	
		(result in $R_{\mathrm{T}}$ )	
		"我们是我们的,我们就是我们的,我们就是我们的,我们就是我们的,我们就是我们的,我们就是我们的。"	

For type  $S_2$ , the instruction MEDR is used in place of MER. Two LER's must be used in place of one.

Operation Operation	<u>Type</u>	<u>Code</u>
<b>+x</b>		No code generated.
	I,I <sub>2</sub>	LACR R <sub>x</sub> , R <sub>x</sub>
	S	LECR R <sub>x</sub> , R <sub>x</sub>
	$\mathbf{s_2}$	LED R <sub>X</sub> , X
		LECR R <sub>x</sub> , R <sub>x</sub>

3.1.2.2 Comparison Operators. The full complement of relational operators is allowed for Integer or Scalar operations between single quantities. Only equal or not equal operators are allowed for arrayed comparisons. No logical variables are created by comparisons. Instead, branching to one of two points is used for true/false relations.

Operation	Type		Code	Alternate Code
X < OP > Y:	I	СН	R <sub>x</sub> , Y	
		вс	COND, not-true-label	
	<b>1</b> 2	c	R <sub>X</sub> , Y	CR R <sub>x</sub> , R <sub>y</sub>
		вс	COND, not-true-label	
	s	CE	R <sub>x</sub> , R <sub>y</sub>	CER R <sub>x</sub> , R <sub>y</sub>
		вс	COND, not-true-label	
	$s_2^{}$	CED	R. Y Company	CEDR R <sub>x</sub> , R <sub>y</sub>
		вс	COND, not-true-label	

Note: For comparisons to the literal 0, the condition code is used directly. If the condition code is not valid, the instruction LR or LER is used to set it.

3.1.2.3 <u>Conversions</u>. Where necessary, conversions are performed in intrinsic or library functions. Some conversions do not require any generation of code.

#### Integer Conversions

Operation	Туре	Code	Alternate Code	3
*I,I, TO S,S,	Ι	LH 5, X		
7,72			argument setup	
	12	<u> Б</u>		
		ACALL atos	actual call	

TYPE OF INTEGER $\alpha$	TYPE OF SCALAR β
Single Precision H	Single Precision E
Double Precision I	Double Precision D

<sup>\*</sup> I TO S does not call library routine; instead code generated is:

I CVFX F<sub>X</sub>, R<sub>X</sub>. 3-21

Operation	<u>Type</u>		Code	Alt	ernate	<u>Code</u>
I,I2 TO BIT	No	code necess	sary			
I TO CHAR	I	LH 5	, <b>x</b>			
		LA 2,	, temp-str	ing-area*		
		ACALL H	TOC			
	1 <sub>2</sub>	<b>L</b> 5	, X			
		LA 2	, temp-str	ing-area*		
		ACALL I	roc			
I TO I	Ι	SRA R	x, 16			
I <sub>2</sub> TO Î	I <sub>2</sub>	SLL R	x, 16			
Scalar Conve	rsions					Andreas (1990) Andreas (1990)
Operation	Type		Code	<u>A1</u>	ternate	Code
s,s <sub>2</sub> TO I,I <sub>2</sub>	g S	LE 0	, x	LER	0, R <sub>x</sub>	argument
	$s_2$	LED 0	, X	LEDR	0, R <sub>x</sub>	setup
		ACALL C	хТОβ			}call
	TYPE OF S	SCALAR α	TYPE OF	INTEGER	β	
	Single Pr	ecision E	Single I	Precision	н	
	Double Pr		Double 1	Precision	I	

S,S, TO BIT	Same as for s	calar to integer
s to char s	LE	0, X
	LA	2, temp-string-area*
	ACALL	ETOC

<sup>\*</sup> temp-string-area contains converted string.

Operation	Type		Code	Alternate Code
	s	LED (	), X	
	4	LA 2	, temp-strir	ng-area*
		ACALL	DTOC	
s to s <sub>2</sub>	s	LE F	$\mathbf{R}_{\mathbf{x}}$ , $\mathbf{x}$	
			R <sub>x</sub> +1, R <sub>x</sub> +1	

3.1.2.4 Assignments. For all assignments, type conversion may take place across the assignment operator. For multiple assignments, the left hand side operands are grouped by data type to minimize the number of conversions performed. The order in which the groups are processed is determined by the following table:

	Right	Hand Ope	erand Typ	е
Left Hand Type Ordering	I	I <sub>2</sub>	S	s <sub>2</sub>
First	I	12	S	s <sub>2</sub>
	<b>1</b> 2	Char	Char	Char
	Char	<b>s</b> <sub>2</sub>	s <sub>2</sub>	S
	s <sub>2</sub>	S	12	1 <sub>2</sub>
Last	S Vector	I -Matrix	1	I

Character is always performed before any right hand side conversion is performed.

The following sequences assume that  $R_{\mathbf{X}}$  has already had the required integer or scalar conversions performed as described in Section 3.1.2.3.

<sup>\*</sup> temp-string-area contains the resultant string.

Operation	Type of Y		Code
y = x;	1*	STH	R <sub>x</sub> , Y
	$\mathbf{r_2}$	ST	R <sub>x</sub> , Y
	S	STE	R <sub>x</sub> , Y
	$\mathbf{s_2}$	STED	R <sub>x</sub> , Y

 $R_{\rm X}$  is also marked as now containing the value Y. Subsequent usages of Y may use this register in lieu of the copy of Y in memory until such time as the contents of this register are destroyed or a label is generated.

\* If X is an integer literal of value 0 or -1, then the following code will be generated:

Y	= 0;		I		ZH	Y
Y	= -1;		I	100	SHW	Y

#### 3.1.3 Bit String Operations

3.1.3.1 Bit String Operators. Bit string operators include the following: AND (&), OR (|), and CAT (||). They generally employ two operands, denoted here as X and Y (of lengths  $N_X$  and  $N_Y$  respectively). X is assumed to be loaded into register  $R_X$  unless otherwise noted. If Y is also in a register, it is represented as  $R_Y$ . Note that the & and | operations will pad the bit length of the shorter bit string to the length of the longer bit string.

Operation	Bit Length	Code	Alternate Code
X & Y	$N_{x}$ , $N_{y} \leq 16$	NR R <sub>x</sub> , R <sub>y</sub>	NHI R <sub>x</sub> , 'Y'*
	$N_x$ , $N_y > 16$	N R <sub>x</sub> , Y	NR R <sub>x</sub> , R <sub>y</sub>
х   ч	$N_x$ , $N_y \leq 16$	OR R <sub>x</sub> , R <sub>y</sub>	OHI R <sub>x</sub> , 'Y'*
	$N_{x}$ , $N_{y} > 16$	O R <sub>x</sub> , Y	OR R <sub>x</sub> , R <sub>y</sub>
x    Y	$N_{y} \leq 16$	SLL R <sub>x</sub> , N <sub>y</sub>	
		OR $R_{\mathbf{x}}$ , $R_{\mathbf{y}}$	
	N <sub>y</sub> > 16	SLL R <sub>x</sub> , N <sub>y</sub>	
		O R <sub>x</sub> , Y	OR R <sub>x</sub> , R <sub>y</sub>

<sup>\*</sup> used only when Y is a BIT literal.

3.1.3.2 Bit String Comparisons. The only possible relational operators for bit strings, as with bit operators, are = or == (see Section 3.3.1.4). The bit strings are padded to be of equal lengths. No logical variables are created by comparisons. Instead, branching to the "not-true-label" occurs with the "not true" condition.

Operation Bit Length Cod		Code	Alternate C		Code	
$X < OP > Y$ $N_X, N_V \le 16$	СН	R <sub>x</sub> , Y		CHI	R <sub>x</sub> ,	'Y'*
	вс		not-tru	e-label		
$N_{x}$ , $N_{v}$ > 16	C	R <sub>x</sub> , Y		CR	R <sub>x</sub> ,	R <sub>y</sub>
	вс	COND,	not-tru	ıe⊷label		

3.1.3.3 Component Subscripting. Component subscripting for bit strings consists of shifting and &'ing out unwanted components of the subscripted bit string. The resultant bit string length,  $N_r$ , determines a binary mask, whose decimal value is  $2^{Nr-1}$ , and bit number "I" of the original bit string is the last component of the resultant bit string.

Operation	Bit Length			<u> </u>	<u>ode</u>
X <sub>subscript</sub>		<b>V</b>		SRL N	R <sub>x</sub> , N <sub>x</sub> -I R <sub>x</sub> , mask**
<sup>X</sup> variable sub		N <sub>x</sub> of Subscript	Forms:	LACR AHI SRL	R <sub>I</sub> , R <sub>I</sub> R <sub>I</sub> , N <sub>x</sub> R <sub>x</sub> , O(R <sub>I</sub> )
	Subscript	Ι	N <sub>r</sub>	N	R <sub>x</sub> , mask**
	3 TO 10 6 AT 11 9 8 AT J K	10 16 9 J + 7 K	8 6 1 8		

3.1.3.4 Bit Conversions. When necessary, conversions are performed in intrinsic or library functions. Some conversions do not require any generation of code.

Used only when Y is a Bit literal. The mask value is equal to  $2^{Nr} - 1$ .

Operation Operation	Bit Length	<u>Code</u>	Alternate Code
BIT TO I		No code necessary	
BIT TO I <sub>2</sub>		LH R <sub>x</sub> , X SRA R <sub>x</sub> , 16	SRA R <sub>x</sub> , 16
BIT TO S,S2	N <sub>x</sub> < 16	LH 5, X ACALL HTOE	LR 5, R <sub>x</sub>
	N <sub>x</sub> > 16	L 5, X ACALL ITOE	LR 5, R <sub>X</sub>
BIT TO CHAR	N <sub>x</sub> < 16	LH 5, X SRL 5, 16	et up of bit-type
	N <sub>x</sub> > 16	L 5, X	set up of bit-type argument
		LA 2, temp-str:  LHI 6, N <sub>X</sub> ACALL BTOC	actual calling sequence
BIT TO CHAR	@ <radix></radix>		HAR except call to as follows:
		<pre><radix> BIN OCT DEC HEX</radix></pre>	routine  BTOC OTOC KTOC XTOC

Temp-string-area contains converted string.

Operation	Bit Length	<u>Code</u>	Alternate Code
BIT TO BIT	$N_x > N_y$ $N_y \le 16$	NHI $R_{x}$ , $2^{N_{y-1}}$	
	$N_{y} > 16$	N R <sub>x</sub> , mask*	

3.1.3.5 Bit Assignments. The following sequences assume that  $R_{\rm X}$  has already had the required conversions performed as described in Sections 3.1.3.3 or 3.1.3.4.

Operation	Length of Bit String Y		Code
Y = X	N < 16	STH	R V**
		-	``x'
	$N_{y} > 16$	ST	R <sub>x</sub> , Y**

If the right hand side of the assignment (X) is a BIT literal as described below, and Ny  $\leq$  16, then the following code is generated:

$$Y = BIN'0';$$
  $N_{Y} \le 16$   $ZH$   $Y$   
 $Y = BIN(16)'1';$   $N_{Y} = 16$   $SHW$   $Y$ 

3.1.3.6 Partitioned Bit Assignments. The following sequences assume that  $R_{\rm x}$  has already had the required conversions performed as described in Section 3.1.3.3 or 3.1.3.4. Definitions of I,  $N_{\rm y}$ , and  $N_{\rm r}$  are as described in Section 3.1.3.3.

Operation	Length of Bit String Y		<u>Code</u>
Y <sub>subscript</sub> =X;	N <sub>y</sub> ≤ 16	LH	R <sub>x</sub> , X
		LH	R <sub>Y</sub> , Y
		SLL	R <sub>x</sub> , N <sub>y</sub> -I
		XR	R <sub>x'</sub> R <sub>y</sub>
		NHI	R <sub>x</sub> , mask***
		XR	$R_{x}$ , $R_{y}$
54 - 12시 (2015 112 112 112 112 112 112 112 112 112 1		STH	R <sub>X</sub> , Y

<sup>\*</sup> The value of the mask is 2 Y-1.

\*\* Note: If  $N_x > N_y$  and  $N_y$  is not exactly 16 or 32, then the following instruction must be added:

\*\*\* Mask: The mask used in a bit store is computed as follows:  $(2^{N_r-1})$ 

In other words, the mask is a sequence of  $N_{\mathbf{r}}$  bits  $shifted \ left \ N_{\mathbf{x}}-I$  bits.

Operation	Length of Bit String Y		Code
Ysubscript=X;	$17 \leq N_{y} \leq 32$	L	Ry' Y
(Con't.)			R <sub>x</sub> , X R <sub>v</sub> , N <sub>v</sub> -I
		XR	R <sub>v</sub> , R <sub>x</sub>
		N XR	Ry, mask* Ry, Rx
			R <sub>y</sub> , Y

If the right hand side of the assignment (X) is a bit literal containing either BIN'0' or BIN(Ny)'l' then if Ny  $\leq$  16 and Y is addressable in SRS format, then the following code is generated:

$$Y_{11 \text{ TO } 13} = BIN'O'; N_{Y} = 16$$

ZB Y, B'111000'

$$Y_{10 \text{ TO } 12} = BIN'111'; N_{Y} = 16$$

SB Y, B'1110000'

If  $N_{V}$  > 16 then the following code is generated:

$$Y_{13 \text{ TO } 20} = BIN'O'; N_Y = 32$$

 $R_{x}$ , =X'FFF00FFF'

NST R<sub>x</sub>, Y

$$Y_{17 \text{ TO } 20} = BIN'111'; N_{Y} = 32$$

 $L = R_{x}' = X'00007000'$ 

OST R, Y

3.1.3.7 Bit Tests.

 $N_x = 1$ 

тн х

IF X<sub>10</sub>

 $N_{x} = 16$ 

BZ <not true label>

TB X, B'1000000'

BZ <not true label>

or

LH R<sub>x</sub>, X

SRL R, 6

NHI R, B'1'

BZ <not true label>

IF ¬X

Same as IF X except BZ changed to BNZ instruction.

<sup>\*</sup> The mask value is computed as:  $(2^{N_{r-1}}) \cdot (2^{N_{x-1}})$ .

#### 3.1.4 Character String Operations

3.1.4.1 Character String Operators. The only character string operator is the CAT (||) operating employing two character string operands denoted here as X and Y (of lengths  $N_X$  and  $N_Y$  respectively). Unless otherwise noted, X is assumed to be loaded into registers  $R_X$ . If Y is also in a register, it is represented as  $R_Y$ .

Operation	Co	<u>de</u>
<b>X</b>     <b>Y</b>	LA	P3, Y
	LA	P2, X
	LA	P1, temp-string-area
	ACALL	CATV

4

3.1.4.2 Character String Comparisons. The full set of relational operators are allowed for character strings (see Section 3.1.1.4 for condition codes). Characters with different lengths are always unequal. No logical variables are created by comparisons. Instead, branching to the "not-true-label" occurs with the "not true" condition.

0perat	<u>ion</u>	Cod	<u>de</u>		
X <op></op>	Ý	LA	P3, Y		
<op> α</op>		LA	P2, X		
= <u>'</u>		ACAL	L CPRa		
<=,   C >=		ВС	COND,	not-true-	label

3.1.4.3 Component Subscripting. Component subscripting for character strings consists of setting the initial,  $N_i$ , and final,  $N_f$ , index values of the subscripted components into registers 5 and 6 respectively, and then branching to the CASP intrinsic.

Operation	<u>C</u>	ođe	Altern	ate Code	
subscripting	LA	P2, X			
Y = X <sub>subscript</sub> ;	LA	Pl, Y			
	LH	5, N <sub>i</sub>			
	LH	6, N <sub>f</sub>	LR	6, 5	if only 1 component
	ACALL	CASP			

3.1.4.4 Character String Conversions. Where necessary, conversions are performed in intrinsic or library functions.

Operation	<u>C</u>	ode
CHAR TO I	LA	2, char
	ACALL	СТОН
CHAR TO I2	LA	2, char
	ACALL	CTOI

Operation Code		
CHAR TO S	LA 2, char ACALL CTOE	
CHAR TO S <sub>2</sub>	LA 2, char ACALL CTOD	
CHAR TO BIT	LA 2, char ACALL CTOB	

CHAR TO BIT @ < radix >

Same as CHAR TO BIT except call to BTOC is replaced as follows:

<radix></radix>	routine
BIN	CTOB
OCT	CTOO
DEC	CTOK
HEX	CTOX

3.1.4.5 Character String Assignments. Either the receiver variable or the assigned variable in a character string assignment may be subscripted. The possible forms are shown below. When subscripting is used, a partitioning of a character string results. The initial element of this partitioned character string is signified by its index: Ni. Similarly the final element has the index Nf. Some examples of HAL/S subscript forms and the resulting Ni and Nf values are:

Subscr	ipt	Form	Ni	Nf
1 TO 3			1	3
5 AT 2			2	6

Operation	<u>Code</u>			
Y =X	LA	P2, X		
	LA	Pl, Y		
	ACALL	CAS*		
Y <sub>subscript</sub> =X	LA	P2, X		
	LA	P1, Y		
	LHI	5, N <sub>iy</sub>		
	LHI	6, N <sub>fv</sub>		
	ACALL	CPAS*		

Operation	<u>Code</u>			
Y=X subscript	LA	P2, X		
	LA	Pl, Y		
	LHI	5, N <sub>ix</sub>		
	LHI	6, N <sub>fx</sub>		
	ACALL	CASP*		
Ysubscript =Xsubscript	LA	P2, X		
	LA	P1, Y		
	LHI	5, N <sub>ix</sub>		
	LHI	6, N <sub>fx</sub>		
	L	7, H'N <sub>iy</sub> , N <sub>fy</sub> '		
	ACALL	CASP		

\* For REMOTE data, CASR is called instead of CAS, CASRP for CASP, etc.

#### 3.1.5 Vector Matrix Operations

3.1.5.1 Vector-Matrix Operators. Vector Matrix operators usually operate on two arguments according to the conventions stated in Section 5.2. Since 3-vectors, and 3x3-matrices have special library routines, their code is listed in the column labeled "3-code", while the code for any other vectors or matrices is listed in the "n-code" column.

Operation	Type	<u>n</u> –	code	<u>3-c</u>	<u>ode</u>	Water St.
V1 + V2	single	LA	P3, V2	LA	P3, V2	
		LA	P2, V1	LA	P2, V1	
		LA	Pl, temp-ar	ea LA	P1, temp	-area
		LHI	5, n	ACALI	. VV2S3	
		ACALI	. vv2sn			

Operation Operation	Type	<u>n-</u>	code	3-00	ode_
V1 + V2	double	LA	P3, V2	LA	P3, V2
		LA	P2, V1	LA	P2, V1
		LA	Pl,temp-area	LA	Pl, temp-area
		LHI	5, n	ACALL	VV2D3
		ACALL	VV2DN		
V1 - V2	to are VV3 VV3S3 (VV3	SN (V)	except that t 73DN for doubl double preci ors respective	e preci sion) i	
-v1	single	LA	P2, V1	LA	P2, V1
		LA	P1, temp-area		Pl, temp-area
			5, n VV7SN	ACALL	VV7S3
-v1	double	routi	as -Vl single nes VV7DN and ize n and size	VV7D3	are called
V1 % V2	single	LA	P3, V2	LA	P3, V2
V1: length n V2: length m		LA	P2, V1	LA	P2, V1
result is nx matrix	m	LA	Pl,temp-area	LA	Pl, temp-area
		THI	5, n	ACALL	V06S3
		LHI	6, m*		
		ACALL	V06SN		
V1 ½ V2	double	that and V	as for single the routines 706D3 for n-ve ectively.	branche	ed to are V06DN
V1 * V2	single	(ille	gal operation	) LA	P3, V2
				LA	P2, V1
				LA	Pl, temp-area
				ACALI	VX6s3
V1 * V2	double	Same VX6D3	as for single is branched	precis to, rat	sion, except that ther than VX6S3.

<sup>\*</sup> If both V1 and V2 are the same size, then this instruction will be: LR 6, 5. 3-33

	Operation	Type	. <u>r</u>	n-code	3-00	<u>ode</u>
	V1 · V2	single	LA	P3, V2	LA	P3, V2
			LA	P2, V1	LA	P2, Vl
			LHI	5, n	ACALL	VV6S3*
			ACALL	vv6sn*		
	V1 · V2	double	that VV6DN	as for single paths routines be and VV6D3 for tors respective	ranched n-vec	d to are
or	M1 + M2 M1 - M2 }	vectors o	f leng	at for adding th equal to th e column size	e proa	uct of the
	V1 M2	single	LA	P3, M2	LA	P3, M2
	V1: length n M2: nxm		LA	P2, V1	LA	P2, Vl
			LA	Pl, temp-area	LA	P1, temp-area
			LHI	5, n	ACALL	VM6S3
			LHI	6, m**		
			ACALL	VM6SN		

<sup>\*</sup> The scalar result of the dot product is left in register F0.

<sup>\*\*</sup> If M2 is of size nxn, then this instruction is: LR 6, 5.

Operation	Type	1	n-code	<u>3-c</u>	ode
VI % WI	double	that VM6DN	as for single the routines h and VM6D3 for he size 3 case	ranche the g	d to are eneral case
Ml ½ Vl Ml: n x m Vl: length m	branched and MV6S3	to are (MV6D	MM except the MV6SN (MV6DN 3 for double plant double plan	for do	uble precision) on) for the
V1 % I*,	single	LE	0, s	LE	0, S
V1 ½ I2*, V1 ½ S		LA	P2, V1	LA	P2, V1
		LA	Pl, temp-area	LA	Pl, temp-area
		LHI	5, n	ACALL	VV4S3
		ACALL	VV4SN		
V1 ½ S2	double	LED	0, S2	LED	0, S2
		LA	P2, V1	LA	P2, V1
		LA	Pl, temp-area	LA	Pl, temp-area
		LHI	5, n	ACALL	VV4D3
		ACALL	VV4DN		
V1/I,V1/I2 V1/s,V1/S2	branched and VV5S3	to are (VV5D	VV5SN (VVSDN	for do	at the routines uble precision) on) for n-vectors
I ½ V1,I2 ½ V S ½ V1,S2 ½ V		Exact	ly the same as	s for V	l ⊭ I, etc.
M1 ½ I,M1 ½ I M1 ½ S,M1 ½ S		the l	as for Vl ½ I ength value sp ct of the row of Ml.	pecifie	d in R5 is the

<sup>\*</sup> Note that in the case of single and double precision integers, they are first converted to scalar form whose value is in F0.

# REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOK

-4610	'n.
- 715	٥.
ala	
	з

Operation	Type	n-code 3-code
M1/I,M1/I2, M1/S,M1/S2		Same as for Vl/I, etc., except that the length value specified in R5 is the product of the row size and the column size of Ml.
I ½ M1,12 ½ M S ½ M1,S2 ½ M		Exactly the same as for VI ½ I, etc., except that the length specified in R5 is equal to the product of the row size and the column size of Ml.
M1**i	single	LHI 6, i Same as for "n-code where n = 3.
(where i is $\epsilon$ a literal or		LA P3, temp-storage-area
constant inte		LA P2, M1
		LA Pl, temp-storage-area
		LHI 5, n
		ACALL MM17SN
M1**i	double	Same as for single precision, except branches to the MM17DN.
M1**0	single	LA P2, M1
		LA P1, temp-storage-area
		LHI 5, n
		ACALL MM15SN
M1**0	double	Same as for single precision, except branches to MM15DN.
M1**T	single	LA P2, M1 LA P2, M1
M1: m x n		LA Pl, temp-stor- LA Pl, temp-stor- age-area age-area
		LA 5, n ACALL MM11S3
시에면 무슨 교육인의 라이는 기술에 함께다		ulio de la companya d LA 6, m
		ACALL MMllsn

Operation	Type	<u>n-c</u>	code	3-00	ode
Ml**T do	ouble	that MM11D	as for single protection that the routine brain or MM11D3 for x 3 matrices re	nched t n x n	to is either matrices
M1 ½ M2 si	ingle	LA	P3, M2	LA	P3, M2
M1: k x m M2: m x n		LA	P2, M1	LA	P2, M1
		LA	Pl, temp+area	LA	Pl, temp-area
		LHI	5, k	ACALL	MM6S3
		LHI	6, m*		
		LHI	7, n*		
		ACALL	MM6SN		
M1 ½ M2 d	ouble	that MM6DN	as for single p the routines br and MM6D3 for he 3 x 3 case r	anched the ge	to are neral case

<sup>\*</sup> Either of the instructions may be of the form: LR 6,5 if n=k, etc.

3.1.5.2 Conditional Operators. The only comparison operators allowed for comparing vectors and matrices are = or ¬ =. Since these comparisons are done on an element-by-element basis, the same routines that are used for size-n vectors are also used for size n x m matrices which are considered to be vectors of length n x m. No logical variables are created by comparisons. Instead, branching to the "not-true'label" occurs with the "not true" condition.

Operation Operation	Type	ņ	ı-code	3-ccd	
V1 <op> V2</op>	single	LA	P3, V2	LA I	93, V2
		LA	P2, V1	LA I	92, V1
		LHI	5, n	ACALL V	VV8S3
		ACALL	VV8SN	BC (	COND, not-true- label
		BC	COND, not-tru	ie-la <del>b</del> e	
V1 <op> V2</op>	double	that t	as for single the routines by V8D3 for n-vectively.	ranched	l to are VV8DN
M1 <op> M2</op>	single	LA	P3, M2	LA	P3, M2
M1, M2: mxn		LA	P2, M1	LA	P2, M1
		LHI	5, mxn	THI	5, 9
		ACALL	vv8sn	ACALL	vv8sn
		вс	COND, not- true-label	ВС	COND, not-true- label
Ml <op> M2</op>	double	Same that	as for single the routine b	precis	ion, except to is VV8DN.

3.1.5.3 Component Subscripting. Possible components of matrices include submatrices, vectors, column vectors, and single components. Possible components of vectors include subvectors and sigle components. The resultant type of component is determined by the subscripts used. Note that double precision operations are not shown - their code is identical except that: a) the called routines will be VVIDN rather than VVISN, etc; b) the index multiplier is 4 instead of 2. Register 7, when used, contains skip values between elements in partitioned matrices (see Section 3.1.1.3).

Operation*		n-code	3-code
$Y = Vx_i;$	LE	$R_{x}$ , $V_{x}$ + 2 * i	N.A.
	STE	R <sub>x</sub> , Y	
$Y = Vx_1;$	LH	R <sub>I</sub> , I	
	LE	R <sub>x</sub> , Vx(R <sub>I</sub> )	N.A.
	STE	R <sub>X</sub> , Y	
$v_{y_I} = x;$	LH	R <sub>I</sub> , I	
	LE	R <sub>X</sub> , X	N.A.
	STE	R <sub>x</sub> , Vy(R <sub>I</sub> )	
$v_{y_n} = v_{x_n}$	AT I		
	LH	R <sub>I</sub> , I	LH R <sub>I</sub> , I
	AR	R <sub>I</sub> , R <sub>I</sub>	AR $R_{I}$ , $R_{I}$
	LA	P2, Vx(R <sub>I</sub> )	LA P2, Vx(R <sub>I</sub> )
	LA	Pl, Vy(R <sub>I</sub> )	LA P1, Vy(R <sub>1</sub> )
	LHI	5, n	ACALL VV1S3
	ACALI	VVlsn	

<sup>\*</sup> i indicates integer literal, I indicates integer variable.

Operation*		n-code		3-code
$M_{V} = Mx_{m AT} I, n A'$	гJ			
assumes $M_{ m V}$ is an $ m m$				
MATRIX *	LH	R <sub>I</sub> , I		<same></same>
	MHI	R <sub>I</sub> , <column of<="" size="" th=""><th>* M<sub>x</sub>&gt;</th><th></th></column>	* M <sub>x</sub> >	
	АН	R <sub>I</sub> , J		
	AR	R <sub>I</sub> , R <sub>I</sub>		
	LA	P2, My (R <sub>I</sub> )		
	<b>L</b>	7, F'delta,0'		
	LA	P1, M <sub>y</sub>		
	LHI	5, m		
	LHI	6, n		
	ACALI	L MM1SNP		
$N_{Y_{\star,I}} = V_{x};$	LH	R <sub>I</sub> , I	LH	R <sub>I</sub> , I
**,I	AR	$R_{I}$ , $R_{I}$	AR	R <sub>I</sub> , R <sub>I</sub>
	LA	P2, V <sub>X</sub>	LA	P2, V <sub>x</sub>
	THI	6, 0	LHI	6, 0
	LHI	7, delta	LHI	7, delta
	LΑ	Pl, M <sub>x</sub> (R <sub>I</sub> )	LA	P1, MX(R <sub>I</sub> )
	LHI ACAL	5, n L VVlSNP	ACA	LL VV1S3P

<sup>\*</sup> i indicates integer literal, I indicates integer variable.

3.1.5.4 Conversions. MATRIX/VECTOR conversions are done by considering matrices as vectors, and assigning the required components to the receiver variable. More than 1 argument requires multiple calls to the vector assign routine (as shown in the second sequence below). Use of double precision operands will cause branches to VVIDN. Otherwise, the code is unchanged.

Operation	<u>n-c</u>	code
VECTOR (M)	LA	P2, M2
Produces vector of size	LA	Pl, temp-area
equal to product of dimension of matrix:		5, nxm VVlsN
$\mathtt{MATRIX}(V_{X},V_{V},V_{Z})$	LA	P2, V <sub>X</sub>
	LA	Pl, temp-area
	LHI	5, n <sub>x</sub>
		VVlsn
	LA	P2, V <sub>V</sub>
		Pl, temp-area+DELTAl
	LHI	5, n <sub>v</sub>
		vvlsn
	LA	P2, V <sub>z</sub>
	LA	Pl, temp-area+DELTA2
	LHI	5, n <sub>z</sub>
	ACALL	VVlsn

<sup>\*</sup> This is an example using several vectors to illustrate the multiple calling of the VVISN (or VVIS3) routine. It applies to the VECTOR shaping function.

3.1.5.5 Assignments. Vectors and matrices may be assigned to other vectors and matrices of the same dimensions. In addition, they may have all elements set to zero by a statement of the form:

$$\overset{*}{M} = 0; \text{ or } \overline{V} = 0;$$

Note that the use of double precision operands will only change the routines branched to: i.e. VV1DN and VV0DN respectively in the code sequences below.

Operation	n-co	<u>đe</u>	<u>3-c</u>	ode
$v_x = v_y$	LA F	v2, v <sub>y</sub>	LA	P2, V <sub>y</sub>
	LA P	ol, v <sub>x</sub>	LA	Pl, V <sub>x</sub>
	LHI 5	$\mathbf{n}$	ACALL	VV1s3
	ACALL V	Vlsn*		
$V_{\mathbf{x}} = 0$	SEDR 0	0, 0	SEDR	0, 0
	LA P	ol, v <sub>x</sub>	LA	Pl, V <sub>x</sub>
	LHI 5		LHI	5, 3
	ACALL V	V0sn**	ACALL	VV0sn
$M_{x} = M_{y}$ and	content	for vectors, ex of register 5 matrix dimension	is equ	that the all to the product
$M_{\mathbf{X}} = 0$				

<sup>\*</sup> For REMOTE data, VRISN is called in place of VVISN.

<sup>\*\*</sup> For REMOTE data, VROSN is called in place of VVOSN.

The temporary area used to store the result of the last HALMAT operation before an assignment can be eliminated if the vector-matrix statement is of a suitable "fa)rm" for optimization and one of four conditions hold. The statement may not have multiple receivers; the single receiver must be a consecutive partition or be nonpartitioned. The precision of the right-hand-side of the statement must match the precision of the receiver. The receiver cannot be a remote variable, and neither the receiver nor the operand(s) of the final HALMAT operation can be name variables, or the terminal of a subscripted structure. Also, variable subscripts on any variables do not allow optimization processing to continue.

Statements that meet these basic requirements can then be checked for the occurrence of a necessary and sufficient condition for optimization. The result of the final operation before the assignment will be stored directly in the receiver if at least one of the following conditions is true:

- 1) a) The receiver is nonpartitioned and the last operation before the assignment HALMAT is a "Class 3" operation. Class 3 operations include matrix-scalar and vector-scalar multiplication and division, vector-matrix addition and subtraction, vector and matrix negation and the built-in function, UNIT.
  - b) The last operation is a "Class 1" operation. The class contains only "matrix raised to 0th power". The result, the identity matrix, can be stored directly in any consecutive receiver.
- 2) The operand(s) are in temporary work areas. Nonconsecutive partitions are moved to work areas when the operands are processed. The result of a previous operation is also in a work area. Operands in work areas are disjoint from the receiver. This is important for "class 2" operations that use the elements of the vector or matrix, vector-vector, and matrix-matrix arithmetic, and matrix transpose and exponentiation (also, the built-in functions, TRANSPOSE and INVERSE). This condition can also hold for class 1 and class 3 operations. If the operation has two operands, both must be in work areas for this condition to be true.

The operand(s) are nonidentical to the receiver. A receiver-operand pair is nonidentical if the operand is in a work area, or if neither variable is a formal parameter and the variables have different symbol table references, or if only one of the variables in a formal parameter and the NEST level of the non-parameterized variable is greater than or equal to the NEST level of the parameterized variable (again, symbol table reference cannot be the same).

```
EXAMPLE1: PROGRAM;

DECLARE MATRIX(3,3), S,T;

PROC: PROCEDURE(A) ASSIGN(B);

DECLARE MATRIX(3,3), A,B,C;

SUBPROC: PROCEDURE(X) ASSIGN(Y);

DECLARE MATRIX(3,3), X,Y,P,Q;

Y2 TO 3,* = X2 TO 3,* + C2 TO 3,*;

B2 TO 3,* = P2 TO 3,* + Q2 TO 3,*;

CLOSE SUBPROC;

CALL SUBPROC(A) ASSIGN(C);

CLOSE PROC;

CALL PROC(S) ASSIGN(T);

CLOSE EXAMPLE1;
```

where

X&Y are parameters, C is not
 NEST\_LEVEL(Y)=2,
 NEST\_LEVEL(C)=1.
Y can be C - cannot assign directly.
P&Q not parameters - ok to assign directly
 NEST\_LEVEL(P)=2,
 NEST\_LEVEL(A)=1.

4) The operand(s) are disjoint with the receiver. A receiveroperand pair can be disjoint in two ways. If the pair is
nonidentical it is, by default, disjoint. If both the receiver
and the operand are consecutively partitioned, they are
disjoint if the partitions do not overlap in any way. If
the receiver and the operand have the same symbol table
reference (are identical) then the two partitions can be
disjoint in either "direction". For example, let A
be a 4-by-4 matrix. Then,

 $A_{1 \text{ TO } 2,*} = A_{3 \text{ TO } 4,*} + \dots$  and  $A_{3 \text{ TO } 4,*} = A_{1 \text{ TO } 2,*} + \dots$  are both disjoint pairs.

If the receiver and operand are possibly identical, then the pair can only be disjoint if all of the operand partition comes after the receiver partition.

EXAMPLE2: PROGRAM; DECLARE MATRIX(6,3), A,D,E; PROC: PROCEDURE (B,C); DECLARE MATRIX(4,3), B,C; Pairs A-B & A-C A<sub>1</sub> TO 2,\* = B<sub>3</sub> TO 4,\* + C<sub>3</sub> TO 4,\*; A<sub>3</sub> TO 4,\* = B<sub>1</sub> TO 2,\* + C<sub>3</sub> TO 4,\*; disjoint Pair A-B not necessarily disjoint CLOSE PROC; CALL PROC( $A_3$  TO 6,\*, $B_3$  TO 6,\*); ( $B_1$  TO 2,\* is really  $A_3$  TO 4,\* =  $D_3$  TO 4,\* +  $E_1$  TO 2,\*; A,D,E are, by default, disjoint because they CLOSE EXAMPLE2; are nonidentical

If the operation has two operands, both receiver-operand pairs must be disjoint for this condition to be true. The non-identical and disjoint checks are made at the same time, so this condition also holds if one pair is disjoint by disjoint partitioning and one pair is disjoint by being nonidentical.

#### 3.1.6 Structure Operations

3.1.6.1 Structure Comparisons. Structure comparisons may only be = or -=. The comparison is done by comparing corresponding terminal elements of the two structure operands in order of their natural sequence. Each terminal element is referenced by adding the displacement of the element to the address of the structure (see Section 3.1.1.3). No logical variables are created. Instead, branching to the "not-true-label" occurs with the "not-true" condition.

Operation	<u>C</u> c	<u>de</u>	
X <op> Y</op>	LA	2, X	
	LA	3, Y	
	LA	2, terminal #1(X)	
	LA	<pre>3, terminal #1(Y)</pre>	
for each	LHI	5, width	
terminal	BAL	4, CSTRUC	
	BC	COND, not-true-la	bel
	•		
	<sam< td=""><td>e for all terminals</td><td><b>&gt;</b></td></sam<>	e for all terminals	<b>&gt;</b>
	ВС	7, true-label	

3.1.6.2 Structure Assignments. The assignment of both major and minor structures is done via the MSTRUC routine. The addresses of the structure nodes being accessed are loaded into registers 1 and 2. The width (in halfwords) of the structure node accessed is loaded into register 5.

Operation	Code	
$\mathbf{Y} = \mathbf{X}$	LA P2, X	
	LA Pl, Y	
	LHI 5, widt	:h
	ACALL MSTRUC	ŧ.

<sup>\*</sup> For REMOTE data, MSTR is called instead of MSTRUC.

#### 3.1.7 Indexing and Arrayed Statements

- 3.1.7.1 <u>Linear Array Indexing</u>. Linear array indexing is the use of subscripts, on an arrayed data type, to produce a one-dimensional resultant array. In the generated code, only one register Ra is needed to keep track of the index value. An initial entry to the array loop (see Section 3.1.7.4), Ra is initialized to a value of 1. On each pass through the loop, Ra is used to define a DELTA value to index the arrayed data (see Section 3.1.1.3). Following this, at the end of the loop Ra is incremented by 1, and is tested to determine if all of the data has been utilized, as described in Section 3.1.7.4. Ra is any available indexing register. Its contents may not be altered during the course of an arrayed statement. If the index in Ra must be shifted to access the word or doubleword data, it must be moved to another register to perform this shift.
- 3.1.7.2 Non-Linear Array Indexing. Non-linear array indexing has more than one index which can change values to produce a multi-dimensional resultant array. The actual code generated, though, can only utilize one register Ra for indexing. Thus, temporary storage is needed to store all but the inner-most index. As with linear indexing, all index values (both in Ra and temporary storage) are initialized to 1. The DELTA value defining the index of each arrayed data item is then computed on the basis of the value of Ra and the index values stored in memory (see Section 3.1.1.3). Following this, each index value is tested against the size of the corresponding dimension (of the resultant array) to determine if all of the data has been utilized and/or which indices are incremented for the next iteration. An example of this is given in Section 3.1.7.4.
- 3.1.7.3 Array Indexing. Arrays may be used in their entirety in HAL/S without explicit subscripting (for example assignment of two equally dimensioned arrays). However, the code generated is very similar to that for non-linear indexing, except that the indicies are tested against the size of the corresponding declared dimensions of the arrays, rather than against the size of the corresponding dimensions of the subscripted array. An example of this is shown in the next section.

## REPRODUCIBILITY OF THE ORIGINAL PAGE IS TOOK

3.1.7.4 Arrayness and Loop Generation. This section has an example of each possible form of array loops, and how indexing is achieved within them. In general, an array loop consists of the following sections:

- a) initialization of index values;
- b) computation of address of array element from index value (see Section 3.1.1.3);
- c) actual operation to be performed on the array element(s) (i.e. assignment, comparison, etc.);
- d) incrementing and testing index values.

It should be noted that non-linear and array indexing produce multiple loops and indices. Since only a single register is available for indexing, temporary storage of index values for outer loops is employed.

Operation Type		Code	<u>Code</u>		
Linear Indexing:		L	7, =H'1,2	0	
$[X] = [Y]_{3 \text{ AT } 2}$	[X]: ARRAY(3) SCALAR	loop:LED	2, Y+4(7)	( _	
	[Y]: ARRAY(5) SCALAR DOUBLE	STE	2, X(7)	}	
			7, loop		

Notes on above example:

- ① initialize
- 2 assignment
- 3 increment and test index

BIX

BIX

L

7, inner-

7, templ
7, outer-

loop

loop

indexing of [I]
 assignment of scalar value to an integer value

initialization and storage of first

initialization of second index value

index value

indexing of [V]

- incrementing and testing second index value
- 7 incrementing and testing first index value

Operation	Type			Code	
Array Indexing:					
[M] = [N]	* [M],[N]: ARRAY(2,3) MATRIX(2,4)	outer-loop:	L ST L	7, =H'l,1' 7, templ 7, =H'l,2'	(1) (2)
		inner-loop:	LH MIH SLL AR	6, temp1 6, ±H'3' 6, 15 6, 7	· ③
			SLL LH MIH AR	6, 3 5, templ 5, =H'3' 5, 7	4
			SLL LA LA ST	5, 3 P2, N(6) P1, M(5) 7, temp2	<b>)</b> (5)
			LHI ACAL	5, 8 L VVlSN	)
			L BIX	7, temp2 7, inner- loop	(6)
Notes on above exact initialization first index va	and storage o	f	L BIX	7, templ 7, outer- loop	} (7)
<ul> <li>indexing of []</li> <li>indexing of []</li> <li>matrix to mat</li> <li>incrementing</li> </ul>		cond index V	alue lue		

### 3.1.8 PROCEDURE/FUNCTION Calls

The PROCEDURE/FUNCTION calling process consists of two parts:

- a) argument set up; and
- b) the actual branch to the subroutine.

Argument set up uses registers 5-7 as needed for passing integers or bit strings, and/or pointers to vectors, matrices, character strings, arrays or structures. Floating point registers 0, 2, and 4 are similarly used to pass scalar arguments. Once all of these registers are utilized, all remaining arguments are placed in a run time stack for the procedure or function.

The actual code generated sets up the arguments in the order that they appear in the HAL/S PROCEDURE or FUNCTION block definition statement. For example, if the function is:

F: FUNCTION(integer\_1, scalar\_1, scalar\_2, vector\_1, integer\_2); then the registers are loaded in the order:

register	7	using LH or	$({f L}, {f C})_{i,j}^{f L}$ and ${f C}_{i,j}^{f L}$ and ${f C}_{i,j}^{f L}$ and ${f C}_{i,j}^{f L}$ and ${f C}_{i,j}^{f L}$
register		using LA to	load the pointer to
		vector 1	
register	F2	using LE or	LED
register		using LE or	LED depending on the
		precision	of scalar l
register	5	using LH or	L depending on the
		precision	of integer_1

Once all arguments are set up, the actual branch is a BAL or SCAL instruction to the CSECT defined for the procedure or function.

A leaf procedure/function is one which has no stack requirements (i.e. no parameters, no stack temporaries, no local addressable data, no ON ERROR statements, and no intrinsic library calls). Such procedures may be called via BAL R4, <routine name>. These routines are exited using BCR 7, R4.

Operation	Args	Co	<u>de</u>	Alte	rnate Code	
Argument Setup	<3 non-scalar	LH	7, arg3	L	7, arg3 or LA	7, arg3
and	<3 scalar	LH	6, arg2	L	6, arg2 or LA	6, arg2
		LH	5, argl	L	5, argl or LA	5, argl
		LE	4, scalar-arg3	LED	4, scalar-arg3	
		LE	2, scalar-arg2	LED	2, scalar-arg2	
		LE	0, scalar-argl	LED	0, scalar-argl	
Actual Call		ACALL	csect-name			
				` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `		
Argument Setup	>3 non-scalar	LH	R, argn			
	>3 scalar	STH	R, stack			
			•	\( (1)	non-scalar	
		LH	R, arg4		into sta	CK
		STH	R, stack	1		
		LE	FR, scalar-argn	): `		
		STE	FR, stack			
		PIE	· Scack	(2)	scalar sto	res into
			•		stack	
		LE	FR, scalar-arg4			
		STE	FR, stack	<i>)</i>		
		LH	5, argl	-	•	
				(3)		
		LE	2, scalar-arg2			
Actual Call		ACALL	csect-name	1		

#### Notes on the above:

- (1), (2)Any additional arguments are generally loaded into any unused register and stored. The actual load op codes may be: L, LH, LA, LE, or LED, depending on the type of argument. Similarly, the stores op codes may be ST, STH, STE, or STED. If the argument already exists in a register, then the code generated will be only a store from that register into the stack.
  - Loading of the first 3 non-scalar, and the first 3 scalar arguments. This is identical to the code shown in the first example above.

## 3.1.9 Block Definition

## 3.1.9.1 PROGRAM and TASK Definition.

Operation

Code

PROGRAM or TASK definition block-name:

LA 0, stack-start\*

LA 1, program-data-csect

STH 1, 5(0)

IAL 0, stack-size

LA 3, local-data-area(1)

STH 3, 9(0)

\* Omitted if SDL option is turned on.

3.1.9.2 PROCEDURE and FUNCTION Definition. Both PROCEDURE and FUNCTION definitions are similar to PROGRAM and TASK definitions. However, floating point store instructions are needed to save any scalar arguments passed via registers.

Operation	<u>Code</u>			<u>A</u> ]	Lternate	Co	<u>de</u>
PROCEDURE or	block-name:						
FUNCTION definition	For COMSUBS {	LA	1,	Program	-data-cs	sect	
	OIILY	STH	1,	5(0)			
		IAL	Ο,	stack-s	ize		
		LA	3,	Local-d	ata-area	a(1)	
		STH	3,	9(0)			
		STE	0,	stack	STED	0,	stack
	optional	STE	2,	stack	STED	2,	stack
		STE	4,	stack	STED	4,	stack

# 3.1.10 Flow of Control Statements

3.1.10.1 <u>IF...THEN...ELSE</u>. The code shown below is for the most general form of the IF...THEN...ELSE statement. It is assumed that the condition code from the conditional expression has been generated (see previous subsections on conditional operations).

Operation Operation		Code
IF <cond exp=""> THEN &lt;&gt; ELSE &lt;</cond>	> BC	cond, else-label
		executable code for THEN clause
	BO	
	else-label:	executable code for ELSE clause
nex	t-statement:	
IF <cond exp=""> THEN &lt;&gt;</cond>	В	c cond, next-statement (executable code for)
		THEN clause
	xt-statement:	

3.1.10.2 <u>DO FOR...Loops</u>. The DO FOR loop has two forms: the iterative, and the discrete. They may also cause termination of the loop by use of the clause UNTIL < >, or WHILE < >. The use of these clauses is shown for the case of the iterative DO FOR forms where the additional code needed has been labeled "UNTIL code" and "WHILE code". This same additional code is generated for the discrete DO FOR and is placed immediately before the executable code within the DO group (the same process as is illustrated with the iterative DO FOR). Note that the code only shows the use of a single precision integer index; double precision integers, and single or double precision scalars follow the same algorithm with the exception that the corresponding full word, or floating point instructions are used when dealing with the index variable.

```
Operation
                                        Code
DO FOR I = a TO b BY c;*
                                    LHI
                                           7, a
                        loop-begin: BC
                                           7, test-label
                                              executable code within
                                                DO group
                                           7, I***
                           repeat**: LH
                                           7, c
                                     AHI
                                           7, I
                        test-label: STH
                                     CHI
                                           7, b
                                     BC
                                           6, loop-begin
                        exit-label:
                                              code for statement fol-
                                                 lowing DO group
```

<sup>\*</sup> Assumes a, b, and c are literal values.

<sup>\*\*</sup> This is referenced by the REPEAT statement (see Section 2.3.10.5).

<sup>\*\*\*</sup> This instruction may be omitted if the REPEAT label is not actually used, and the loop index I is already in the designated register.

```
Code
      Operation
                                                          UNTIL code
                                      ZH
                                            temp-area
DO FOR I = a TO b BY c
           UNTIL <cond exp>;
                                      LHI
                                            7, a
                                            7, test-label
                                      BC
END;
                                            temp-area
                        loop-begin:
                                      TS
                                            4, first-state-
                                      BC
                                               ment*
                                                                UNTIL code
                                             cond for exp.
                                            cond, exit-label
                                      BC
                                             executable code
                   first-statement:
                                             within DO group
                          repeat**:
                                      LH
                                             7, I
                                             7, c
                                       AHI
                        test-label:
                                       STH
                                             7, I
                                       CHI
                                             7, b
                                       BC
                                             6, loop-begin
                                              code for statement
                        exit-label:
```

<sup>\*</sup> This is done to avoid testing the <cond exp> until after executing through the loop at least once.

<sup>\*\*</sup> This is referenced by the REPEAT statement (see Section 3.1.10.5).

<sup>\*</sup> This is referenced by the REPEAT statement (see Section 3.1.10.5).

Operation		<u>Code</u>
DO FOR I = 11 TO 12 BY 13	LH	5, 12
	STH	5, temp-test
END;	LH	6, 13
(Il, I2, I3: variables)	STH	6, temp-incr
	LH	7, I1
	BC	7, test-label
loop-begin:	•	executable code within DO group
repeat*:	LH	7, I
	AH	7, temp-incr
test-label:	STH LH	7, I 5, temp-incr
	LA	5, loop-begin
	вс	5, positive-test**
	СН	7, temp-test
	BCR	5, 5
	BC	7, exit-label
positive-test:	СН	7, temp-test
	BCR	6, 5
exit-label:	•	code for statement following DO group

<sup>\*</sup> Repeat label (see Section 3.1.10.5)

<sup>\*\*</sup> This branch is determined by the condition code set by the previous LH 5, temp-incr instruction.

\*

3.1.10.3 DO WHILE/UNTIL. Both of these forms of DO groups are essentially the same except that the DO UNTIL does not test its conditional expression until it has finished executing the code once. In both cases, the condition is tested as detailed in preceeding subsections.

<u>Operation</u>		Code
DO WHILE <cond exp=""> repeat</cond>	•	code for conditional expression
	BC	cond, exit-label
		code for statements within DO group
	BC	7, repeat
exit-label:		code for statement following DO group
DO UNTIL <cond exp=""></cond>	вс	7, first-statement
repeat:		code for conditional expression
	вC	c cond, exit-label
first-statement:		code for statements within DO group
	вс	7, repeat
exit-label:		<pre>code for statement following DO group</pre>
计加工 海上的 医环状腺 化二氢氯化甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基		

3.1.10.4 DO CASE. The DO CASE statement is used to select one of a collection of statements for processing.

### Code Operation R<sub>C</sub>, I DO CASE I; LH 6, else-case-label ① <statement 1> <statement 2> 2, case-vector LA <statement n> $R_{C}$ , 0(2) CH END: 1, else-case-label BC 4, 0(R<sub>C</sub>, 2) LH BCR 7, 4 else-case-label: <else statement code> 7, exit-case-label <statement 1> 7, exit-case-label <statement 2> 7, exit-case-label <statement n> exit-case-label: Data H'n' DC case-vector Y(statement 1) DC DC Y(statement 2) Y(statement n) DC

① bounds checks on case number. Omitted if ELSE case not specified.

3.1.10.5 GO TO, REPEAT, EXIT. All of these statements take the form of unconditional branches. It should be noted that REPEAT and EXIT statements may only be used inside DO groups. See Sections 3.1.10.2 and 3.1.10.3 for the locations of the "repeat" and "exit-label" within a DO group.

Operation		Code	
GO TO label	BC	7, label	
REPEAT label	BC		"repeat" is the location of the code which determines whether DO group iteration is finished or not.
EXIT	вс	7, exit-label	"exit-label" is the location of the code
EXIT label			immediately following the end of the DO group.

3.1.10.6 RETURN. The RETURN statement will branch back from the code for a function to the code immediately following the function's invocation.

Operation			Cod	<u>de</u>				
Procedures & Functio	ns							
RETURN		SRET	7, 0	)		ormal		
		BCR	7, 4	1	1	eaf proc function	edure	or

Programs & Tasks

RETURN

SVC =H'21'

# 3.1.10.7 ON ERROR/OFF ERROR/SEND ERROR.

## Operation Code ON ERROR :m < stmt> LA 4, <stmt> STH 4, error table entry 1. LHI 4, <action>\* STH 4, error table entry 0 7, next-statement <stmt>: <code for stmt> next-statement: : } code for next statement SIGNAL ON ERROR N:m SYSTEM[AND SET <event>] RESET LA 4, <event> only if event action phrase present 4, <action> LHI 4, error table entry 0 SIGNAL ON ERROR IGNORE [AND SET <event>] RESET STH 4, error table entry 1 only if event action phrase LHI 4, <action>\* STH 4, error table entry 0 SEND ERROR n:m SVC = X'0014nnmm' OFF ERROR<sub>n:m</sub> ZH error table entry 0

<sup>&</sup>lt;action> contains action code, error code, and error group as defined in HAL/FCOS ICD.

#### 3.1.11 Built-In Functions

3.1.11.1 Inline Built-in Functions. The following built-in functions emit the inline code shown in the following sequences. In all cases, it is assumed that  $R_{\rm X}$  contains the argument except when a specific load instruction is shown. The results will always be in register  $R_{\rm Y}$ .

5110	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	- 3	<b>Y</b>
Operation	Type		Code
ABS (arg)	scalar, single	LE	R <sub>y</sub> , arg
		LECR	R <sub>y</sub> , R <sub>y</sub>
		BC	2, *-1
	scalar, double	LED	R <sub>y</sub> , arg
		LECR	$R_{y}^{y}$ , $R_{y}$
		BC	Ry, Ry 2, *-1
	integer, single	LH	R <sub>y</sub> , arg
		LACR	Y R., R.
		вс	R <sub>y</sub> , R <sub>y</sub> 2, *-1
	integer, double	L	R <sub>y</sub> , arg
		LACR	Ry, Ry
		BC	2, *-1
LENGTH (char)	character string	LH	R <sub>y</sub> , char
		NHI	R <sub>y</sub> , 255
SIGN(arg)	scalar, single	LE	$R_{\mathbf{x}}$ , arg
			R <sub>Y</sub> , 1
			R <sub>x</sub> , R <sub>x</sub>
and the first of the second of			5, continue
		LECR	R <sub>y</sub> , R <sub>y</sub>
	continue		

Operation Operation	Type	Coc	<u>le</u>
	scalar, double	LED	R <sub>x</sub> , arg
		LED	Ry' D'41100000000000000
		LEDR	$R_{x}^{I}$ , $R_{x}$
			5, continue
		LECR	Ry' Ry
	continue:		
	integer, single	LH	R <sub>x</sub> , arg
			R <sub>Y</sub> , 1
		LR	R <sub>x</sub> , R <sub>x</sub>
			5, continue
		LACR	R <sub>V</sub> , R <sub>V</sub>
	continue:		
		:	
	integer, double	L	R <sub>x</sub> , arg
		L	$R_{y'} = F'1'$
		LR	R <sub>x</sub> , R <sub>x</sub>
		вс	5, continue
		LACR	$R_{\mathbf{y}'}$ , $R_{\mathbf{y}}$
	continue		
		•	
SIGNUM(arg)	scalar, single		R <sub>x</sub> , arg
		LFLI	$R_{\mathbf{Y}}$ , 1
		LER	$R_{\mathbf{X}}$ , $R_{\mathbf{X}}$
		вс	
		BC	4, equal
		LECR	Ry, Ry
		вс	7, continue
	equal:	SER	$R_{v}$ , $R_{v}$
	continue		

Operation	Type	Cod	<u>le</u>
	integer, single	LH LFXI	R <sub>x</sub> , arg R <sub>y</sub> , 1
		LR BC BC	R <sub>x</sub> , R <sub>x</sub> 1, continue 4, equal
		BC	Ry, Ry 7, continue
	equal: continue:	SR :	R <sub>y</sub> , R <sub>y</sub>
	integer, double	L L	R <sub>x</sub> , arg R <sub>y</sub> , =F'1'
		LR BC	R <sub>x</sub> , R <sub>x</sub> 1, continue
		BC LACR BC	A, equal  R, R Y  7, continue
	equal continue		Ry, Ry
SUBBIT <sub>m TO n</sub> (arg)	<pre>integer, single, or bits of length &lt; 16</pre>	SRL NHI	R <sub>y</sub> , 16-n R <sub>y</sub> , mask*
SUBBIT <sub>m-n+lAT m</sub> (arg	) integer double, or bits of length > 16, or scalar single	SRL N	R <sub>y</sub> , 32-n R <sub>y</sub> , F'mask'*

<sup>\*</sup> The mask value is:  $2^{(n-m+1)}-1$ .

<u>Operation</u>	Туре		Code
SHL (arg, n)	integer	SLL	Rarg, n
SHR (arg, n)	integer	SRA	Rarg, n
XOR(X,Y)	Bit, n<16	LH	R <sub>y</sub> , Y
		XR	R <sub>x</sub> , R <sub>y</sub>
	Bit, n>16	x	R <sub>X</sub> , Y
		or XR	R <sub>x</sub> , R <sub>y</sub>
MIDVAL(X,Y,Z)	scalar	LE	F0, X
		LE	F1, Y
		MVS	FO, Z

3.1.11.2 Out of Line Functions. Out of line functions require branches to the run time library.

The registers needed for parameter passing, and the name of the library routine branched to, are specified in the tables of Section 5. Examples are given for representative argument types.

<u>Operation</u>	Type of X	Cod	<u>le</u>
cos(x)	scalar, single	LE ACALL	
SQRT (X)	scalar, double	LED ACALL	
ABVAL (X)	vector(3), double	LA ACALL	2, X VV9D3
TRANSPOSE (X)	<pre>matrix(m,n), double</pre>	LA LA	Pl, temp-area 5, m
	<pre>matrix(3,3), single</pre>	LA	P2, X P1, temp-area MM11S3
UNIT(X)	vector(3), single		P2, X P1, temp-area VV10S3
RANDOMG		ACALL	RANDG
TRIM(X)	character	LA	P2, X P1, t.emp-area CTRIMV
MAX (X)	array(n)	LA LHI ACALL	5, n

3.1.11.3 Shaping Functions. Shaping functions are explicit invocations of type conversion. The generated code for shaping functions has been described in previous subsections where conversions have been described (see Sections 3.1.2.3, 3.1.3.4, 3.1.4.4, and 3.1.5.4).

In addition, when conversion functions are used in a true "shaping" sense, (e.g. MATRIX(<integer array>)), a subroutine is used to move contiguous elements, with possible conversion, to a result location of the desired shape.

#### Example:

MATRIX(A) where A is a 9 element integer array

LA P2, A

LA Pl, <result loc>

LHI 6, X'0002' flags\*

LHI 5, 9 size

ACALL QSHAPQ

Values: 0 = H

1 = I

2 = E

3 = D

<sup>\*</sup> Flags: 1st 8 bits indicate input data type.
2nd 8 bits indicate output data type.

## 3.1.12 Real Time Statements

All REAL TIME statements are implemented by means of a supervisor call (SVC) instruction which has as its address a pointer to a parameter list. The first halfword of this parameter list contains a number which identifies the type of real time call. The remainder of the parameter list varies with the service being requested.

The specific forms of the SVC parameter lists are those described in the HAL/FCOS ICD document.

For real time statements in non-REENTRANT blocks, the SVC parameter lists are in the block's data area. Any invariant portions of the parameter lists are implemented by initialized data. Parts of the parameter lists which are runtime-dependent are created by execution of in-line code preceeding the SVC instruction.

For real time statements in REENTRANT blocks, the SVC parameter lists are dynamically created in the stack by executable code preceeding the SVC instruction.

3.1.12.1 WAIT Statement. The WAIT statement may use registers 0, 1 to contain a double precision time value specified in seconds. If the UNTIL option is specified, the time value is expressed as mission elapsed time. Any other times are 'delta-time' from the current mission elapsed time. If a time value is not specified in the WAIT statement, then the registers will not be affected.

Operation  WAIT n  n: literal  SVC  parameter-list  WAIT X  X: scalar double  WAIT FOR DEPENDENT  WAIT FOR X  X: event value  WAIT UNTIL X  X: scalar double  LED  0, X  SVC  parameter-list  SVC  parameter-list  VAIT FOR X  X: event value  SVC  parameter-list  LED  0, X  SVC  Parameter-list  WAIT UNTIL X  X: scalar double  LED  0, X	Cite regree			Cođe
WAIT N N: Itterat of n'  SVC parameter-list  WAIT X X: scalar double LED 0, X  SVC parameter-list  WAIT FOR DEPENDENT SVC parameter-list  WAIT FOR X X: event value SVC parameter-list  WAIT UNTIL X X: scalar double LED 0, X	Operation	<u>Туре</u>		
WAIT X X: scalar double LED 0, X SVC parameter-list WAIT FOR DEPENDENT SVC parameter-list WAIT FOR X X: event value SVC parameter-list WAIT UNTIL X X: scalar double LED 0, X	WAIT n	n: literal	LED	
WAIT X X: SCATAL GOADS  SVC parameter-list  WAIT FOR DEPENDENT SVC parameter-list  WAIT FOR X X: event value SVC parameter-list  WAIT UNTIL X X: scalar double LED 0, X			SVC	parameter-list
WAIT FOR X X: event value SVC parameter-list WAIT UNTIL X X: scalar double LED 0, X	X TIAW	x: scalar double		
WAIT FOR X X: event variation of the control of the	WAIT FOR DEPENDI	ENT	SVC	parameter-list
WAIT UNTIL X X: SCAIAL GOODES	실리 생각이 하셨다면 하였다.		svc	parameter-list
그 많은 그는 그림 등을 가지 하는 문장이 되었다. 중에 가는 것은 하늘이 하는 것인데 그는 것은 그리고 하는 것이다. 그리고 말하는 것이다.	WAIT UNTIL X	X: scalar double	LED SVC	0, X parameter-list

# 3.1.12.2 CANCEL, TERMINATE Statements.

Operation

Code

CANCEL
CANCEL<task id>
TERMINATE
TERMINATE<task id>

SVC parameter-list

# 3.1.12.3 SIGNAL, SET, RESET Statements.

Operation

Type

1atched

Code

SIGNAL<event var>
SET<event var>

RESET<event var>

latched or unlatched latched

SVC parameter-list

## 3.1.12.4 UPDATE PRIORITY Statement.

Operation

Type

Code

UPDATE PRIORITY TO i
UPDATE PRIORITY<taskid> TO i

SVC parameter-list

3.1.12.5 SCHEDULE Statement. In the following code generation sequences, a schematic representation of possible SCHEDULE statement forms has been used. The symbol [] means that one of the contained elements may appear in the statement form without affecting the generated code. The symbol {} means that one of the contained elements must be included in the statement form - but which one does not affect the code generated.

In general, the code differs only when time values are specified in the SCHEDULE statement. This requires that the time values be specified in double precision format in certain registers as shown below.

### Operation

Code

SCHEDULE<label>[ON<event exp>]PRIORITY(I)[DEPENDENT][WHILE<event exp>]

SVC parameter-list

SCHEDULE < label > {AT X } PRIORITY (I) [DEPENDENT] [WHILE < event exp > ]

LED 0, D'X'

SVC parameter-list

SCHEDULE < label > [ON < event exp > ] PRIORITY (I) [DEPENDENT], REPEAT { AFTER X } EVERY X }

LED 2, D'X'

SVC parameter-list

SCHEDULE < label > [ON < event exp > ] PRIORITY (I) [DEPENDENT] UNTIL X

LED 4, D'X'

SVC parameter-list

[WHILE<event exp>]
[UNTIL<event exp>]

LED 0, D'X'

LED 2, D'Y'

SVC parameter-list

3 - 72

 ${\tt SCHEDULE < label > \{ \substack{ \texttt{AT} & X \\ \texttt{ON} & X } \} \texttt{PRIORITY(I) [DEPENDENT]UNTIL Y} }$ 

LED 0, D'X'

LED 4, D'Y'

SVC parameter-list

LED 2, D'X'

LED 4, D'Y'

SVC parameter-list

UNTIL Z

LED 0, D'X'

LED 2, D'Y'

LED 4, D'Z'

SVC parameter-list

#### 3.1.13 I/O Statements

3.1.13.1 <u>Initiation</u>. Initiation of either READ, READALL, or WRITE statements consists of a branch to the IOINIT library routine. Register 1 contains the I/O channel number, and register 0 indicates the type of I/O to be initiated.

Operation	Type	<u>c</u>	ode
READ(n)		LHI	6, n 5, 0
		ACALL	IOINIT
READALL(n)		LHI LHI	6, n 5, 1 IOINIT
WRITE(n)		LHI LHI ACALL	6, n 5, 3 IOINIT
		•	

3.1.13.2 <u>Input.</u> In all cases, the code sequences below follow the I/O initiation process described in the previous subsection. It is assumed that any conversions have been done previous to the code sequences shown; the resultant type determines which type of code sequence is generated. Note that vector and matrix partitioning require that the first element of the partition be known; additionally, matrices require a DELTA value to be known to skip over those elements (in the "natural sequence") which are not part of the resulting partitioned matrix (see Section 2.1.1.3).

Operation	Type	Cod	<u>le</u>
READ(), I,	integer, single	:}	initiation
		LA 2, I ACALL HIN	
	integer, double	• }	initation
		LA 2, I ACALL IIN	
READ(),, S, 111	scalar, single	$\cdot$	initiation
		LA 2, S ACALL EIN	
	scalar, double	<b>:</b> }	initiation
		LA 2, S ACALL DIN	
READ( ), V, V	ector(n); single		initiation
		LA 2, V XR 7, 7 LHI 5, 1 LHI 6, n ACALL MM20	SNP

	Operation Operation	Type	Code
	READ(), V,	partitioned vector of length n whose first element is located at 'V+ displacement'	initiation  LA 2, V+displacement  XR 7, 7  LHI 5, 1  LHI 6, n  ACALL MM20SNP
		<pre>vector(n); double (partitioned or not partitioned)</pre>	same except branches to MM20DNP
	READ(), M,	<pre>matrix(m,n); single</pre>	initiation
			LA 2, M XR 7, 7 LHI 5, m LHI 6, n ACALL MM20SNP
	READ(), M,	partitioned matrix whose resultant size is mxn, first element is M+displacement.	initiation  LA 2, M+displacement  LHI 7, DELTA  LHI 5, m  LHI 6, n  ACALL MM20SNP
		<pre>matrix(m,n); double (partitioned or not partitioned)</pre>	Same except branches to MM20DNP
or	READ(), C, READALL(),C,1	character string	initiation  LA .2, C ACALL CIN
or	READ(),  C <sub>m</sub> TO n'  READALL(),  C <sub>m</sub> TO n'	partitioned character string	initiation  LA 2, C  LHI 5, m  LHI 6, n  ACALL CINP

	^	Y
2	)	

Operation	Type	Code	
READ(), $C_n$ , or READALL(), $C_n$ ,	single partitioned character string	•	initiation
		LA 2,C LHI 5,n LR 6,5 ACALL CINP	
READ(), B,	bit string(of length n)		initiation
		LHI 6, n ACALL BIN* ST 6, B	
Arrayed Input	The actual code gener		

type of array. Thus, the code will consist of an array loop (see Section 2.1.7.3) which contains the proper code for inputting of each array element using the code shown above (corresponding to the array element type).

3.1.13.3 Output. In all cases, the code sequences below follow the I/O initiation processes described in Section 2.1.12.1. It is assumed that any conversions have been done previous to the code sequences shown; the resultant type determines which type of code sequence is generated. Note that vector and matrix partitioning require that the first element of the partition be known; additionally, matrices require a "delta" value be known to skip over those elements (in the "natural sequence") which are not part of the resulting partitioned matrix.

Operation Operation	<u>Type</u>	<u>Code</u>	
WRITE( ), I,	integer, single		initiation
		LH 5, I ACALL HOUT	Intration
	integer, double	•}	initiation
		L 5, I ACALL IOUT	

<sup>\*</sup> BIN returns the bit string input in register R6.

Operation	<u>Type</u>	Code
WRITE( ), S,	. scalar, single	initiation
		LE 0, S ACALL EOUT
	scalar, double	LED 0, S ACALL DOUT
WRITE( ), V,	vector(n); single	: } initiation
		LA 2, V XR 7, 7 LHI 5, 1 LHI 6, n ACALL MM21SNP
WRITE(), V,	partitioned vector of length n whose first element is located at 'V+displacement'	initiation  LA 2, V+displacement  XR 7, 7  LHI 5, 1  LHI 6, n  ACALL MM21SNP
	<pre>vector(n); double (partitioned or non-partitioned)</pre>	Same except branches to MM21DNP
WRITE(), M,	matrix(m,n); single	initiation  LA 2, M  XR 7, 7  LHI 5, m  LHI 6, n  ACALL MM21SNP
WRITE(), M,	partitioned matrix of resultant size mxn whose first elemen is M+displacement	initiation  LA 2, M+displacement  LHI 7, delta  LHI 5, m  LHI 6, n  ACALL MM21SNP

Operation	Type	Code
	<pre>matrix(m,n); double (partitioned or not  partitioned)</pre>	same except branches to MM21DNP
WRITE(), C,	character string	initiation
		LA 2, C ACALL COUT
WRITE(), C <sub>m</sub> TO n	partitioned character string	$\left. \begin{array}{c} \cdot \\ \cdot \end{array} \right\}$ initiation
		LA 2, C LHI 5, m LHI 6, n ACALL COUTP
WRITE(), C <sub>n</sub> ,	single partitioned character string	initiation  LA 2, C LHI 5, m
		LA 2, C LHI 5, m LR 6, 5 ACALL COUTP
WRITE(), B,	bit string (of length n)	: } initiation
		L 5, B LHI 6, n ACALL BOUT
Arrayed Output	of array. Thus, the c loop (see Section 2.1. outputting of each arr	ted depends on the type ode will consist of an array 7.3) to cause iterative ay element using the code ding to the array element

### 3.1.14 NAME Operations

# 3.1.14.1 NAME Comparisons. NAME comparisons may only be or --.

Operation		<u>Code</u>
NAME (X) <op> NAME (Y)</op>	LH	R <sub>v</sub> , X
X,Y - NAME Variables	LH	R <sub>V</sub> , Y
	CR	$R_{\mathbf{x}}^{1}$ , $R_{\mathbf{v}}$
	BC	COND, not-true-label
	BC	7, true-label
NAME(X) <op> NAME(Y)</op>	LA	$R_{x}$ , $x$
X - declared variable Y - NAME variable	LH	TORNOR Y
	CR	$R_{\mathbf{x}}^{1}$ , $R_{\mathbf{v}}$
	BC	COND, not-true-label
	BC	7, true-label

3.1.14.2 NAME Assignments. The variable Y in the following examples may only be a NAME variable. The variable X may be either an actual or NAME variable having declared properties identical to Y.

NAME (Y) =			LA	R <sub>v</sub> ,	X
where X	is declared	variable	STH		
NAME (Y) =	NAME (X);		LH	R.	х
where X	is NAME vari	lable		R <sub>x</sub> ,	

#### 3.1.15 %MACROS

The following %MACROS are recognized by the HAL/S-FC compiler and produce the indicated code.

### 3.1.15.1 %SVC.

Operation		Code
%SVC (a)		svc a

3.1.15.2 % NAMECOPY. This operation works in the same manner as NAME assignments except that the operands must be structures, but not necessarily having identical properties.

Operation Operation	Code	2	
	LA	Ř,	X
where X is actual variable	STH	R <sub>x</sub> ,	Y

## 3.1.15.3 %COPY.

<u>Operation</u>	cation Code		
%COPY(X,Y)	L	R <sub>x</sub> ,	=Y(X, size of Y)
	L	R <sub>v</sub> ,	=Z (Y)
	MVH	-	
%COPY(X,Y,n)	L	R <sub>x</sub> ,	=Y(X,n)
	L	R <sub>v</sub> ,	=Z (Y)
	MVH	_	
%COPY(X,Y,5);	L	R <sub>v</sub> ,	<b>Y</b>
	ST	R <sub>v</sub> ,	<b>X</b>
		$R_{v}$ ,	
		R <sub>v</sub> ,	
	LH		Y+4
	STH	Ry,	

#### 3.1.16 NONHAL References

Definition and use of the NONHAL construct in the HAL/S-FC compiler system results in an unimplemented feature message from the code generator.

## 3.2 Object Code Naming Conventions

Each successful HAL/S compilation produces several named control sections (CSECTs). The CSECT names are derived according to the following rules:

- a) HAL/S compilation unit names are transferred to the emitted object code by using only the first six characters of the HAL/S name. The name will be padded or truncated to six characters where necessary.
- b) Any occurrence of the underscore character () in the first six characters of a PROGRAM, PROCEDURE, FUNCTION, TASK, or COMPOOL is eliminated. The resulting characters are joined together to produce the characteristic name of the compilation unit (e.g. A B C becomes ABC). Additional characters are placed on the front of the resultant name to form the final name for each of the individual situations in which the name is used. All CSECT names therefore take the form:

#### CCNNNNNN

where the value of cc for individual cases is:

PROGRAM ; \$0

TASKS : \$c c=(1-9, A-2)

COMSUBs #C

Internal procs : an a=(A-Z), n=(0-9)

DECLAREd data : #D
COMPOOL : #P

Process Directory Entries: #E

Z-con to comsub : #Z

Remote data : #R

Exclusive data : #X

In addition to CSECT's produced by the compiler, the HAL/S-FC system defines other CSECT's, some of which are referenced by compiler-emitted code. These CSECT types and their naming conventions are:

Z-con to library routine: #Q

Data for library routines: #L

## 3.3 Printed Data From Phase II

Under control of the LIST compiler option, Phase II will produce a formatted, mnemonic listing of the object code produced for the compilation unit. In addition to the assembler-type mnemonic instruction listing, a full hexadecimal listing of the emitted code is also produced.

This object code listing is normally appended to the Phase I primary source listing as defined by the SYSPRINT DD card. However, use of the SDL compiler option in addition to the LIST option causes the object code listing to be produced through the OUTPUT7 DD card. The listing thus produced is compatible with the ABSLIST function of the AP-101 Link Editor. The HAL/SDL ICD contains the detailed description of the ABSLIST format.

### 3.4 Symbol Table Augmentation

Phase II inherits an initialized symbol table from Phase I. In the course of generating code, Phase II makes additions to the symbol table which are inherited, in turn, by Phase III. These additions are generally in the area of data addressing.

Information is added in two of the symbol tables parallel arrays:

- The SYT\_ADDR array is filled with data offset information indicating the relative location of data items within CSECTs.
- The EXTENT array is filled with information about the size of the storage allocated to individual data items.



### 3.5 Statement Table Augmentation

Phase III inherits, in a secondary storage device, the statement table produced by Phase I. If the ADDRS compiler option is in effect, Phase I leaves room in the statement table for beginning and ending addresses of individual HAL/S statements. This information is filled in by Phase II after the generation of the executable code has been performed. The completed statement table is then left for use by Phase III.

## 4.0 PHASE III - SIMULATION DATA FILE GENERATION

Phase III of the HAL/S-FC compiler has the primary function of providing Simulation Data Files (SDFs) for each unit of compilation. Phase III also produces user-oriented printouts upon special request. This section deals with the following Phase III functions:

- SDF generation
- Printed data

### 4.1 SDF Generation

Phase III synthesizes the SDF for a compilation unit from data received from previous Phases of the compiler. This data is primarily in two areas: a) The symbol table, created by Phase I and augmented by Phase II, and b) The statement table similarly created by Phase I and II.

The detailed format of an SDF is controlled by the HAL/SDL Interface Control Document. The reader is referred there for details of SDF design beyond the overview presented in the next section.

#### 4.1.1 Overall SDF Design

A Simulation Data File (SDF) is an organized and directoried collection of block, symbol, and statement data which is created by the HAL compiler from a single unit of compilation and stored in a permanent form for later use by simulation processors.

There are basically three types of information contained in an SDF. These are:

1) Symbol Data - contains the attributes of HAL symbols (labels and variables) such as name, class and type, relative core address, number of bytes in core occupied, etc. Also contains arrayness and dimensionality for arrayed variables, template linkages for elements of structures, and cross-reference information listing all statements within the compilation unit that may assign values to the symbol.

- 2) Statement Data contains the attributes of HAL statements such as type, Statement Reference Numbers (SRNs) if specified by the user, indices for all labels attached to each statement, and indicies for all variables which may be assigned values by that statement. Also may optionally contain the relative core addresses of the first and last executable instructions emitted for that statement.
- Block and Directory Data contains information about each HAL block and the symbols and statements contained within that block, plus information concerning the layout and organization of the SDF which minimizes the time needed to access desired data entries.

An SDF is produced for all compilation units unless suppressed by the user (the TABLES/NOTABLES option). In the case of COMPOOL compilations, the SDF becomes somewhat simplified, having no executable statements and, consequently, no cross-reference data for its symbols.

SDFs are created as members of Partitioned Data Sets (PDSs) and are assigned names of the form ##CCCCCC, where CCCCCC is the first six characters of the compilation unit name with any and all underscore characters removed. (Example: the SDFs for the compilation units SAMPLER and TEST\_SAMPLE would be assigned the names ##SAMPLE and ##TESTSA, respectively). The members are written in fixed record format with a block size and logical record length of 1680.

The structure of the SDF will support three efficient types of access:

- 1) Given the name of a symbol, and the name of the block in which it was declared, obtain the attributes of the symbol.
- 2) Given a Statement Reference Number (SRN), obtain the attributes of the statement.
- 3) Given an Internal Statement Number (ISN), obtain the attributes of the statement.

In access methods 1) and 2), the SDF directory plays a key role. When the symbol name and its block are given, the directory will identify which particular physical record of the SDF contains the corresponding fixed-length Symbol Node. Once this record has been read into core, a simple and fast binary search will locate the symbol node which in turn "points" directly to the attributes of the symbol which are contained within a variable-length Symbol Data Cell. A virtually identical procedure can be used to locate statement data when the SRN is given. In this case, the fixed-length nodes involed in the binary search are called Statement Nodes, and their corresponding variable-length data cells are called Statement Data Cells.

In contrast to access methods 1) and 2), which require directory help followed by binary searches, method 3) is direct. This is because there is a one-to-one correspondence between the ISN (compiler-generated Internal Statement Number) and the order of the Statement Nodes. The HAL/SDL ICD contains detailed descriptions of the SDF organization.

## 4.2 Phase III Printed Data

For each invocation of Phase III, a set of tabular data is printed. The information presented deals with parameters relating to the SDF produced, such as number of SDF pages, numbers of block, symbol, and statement nodes, etc.

In addition to the information which is always printed, two optional printouts are available. Under control of the TABLST compiler option, the user may request that symbolic, structured dump of the SDF be provided. In addition, under control of the TABDMP compiler option, the user may request that the contents of the SDF be displayed in a hexadecimal format, page by page.

#### 5.0 RUN TIME LIBRARY

### 5.1 Introduction

This section describes the HAL/S-FC runtime library as used to support the HAL/S-FC compiler. The material is organized to present both general design concepts and detailed interface and algorithm information. Following an introductory discussion of general conventions used throughout the library, descriptions of the individual routines are grouped according to the basic type of the routine. Each group is introduced by a quick-reference chart containing basic interface data.

## 5.2 Basics and Conventions

#### 5.2.1 Origin and Format

The HAL/S-FC compiler comes supplied with a run time library. The library is a partitioned dataset (PDS) in IBM AP-101 load module format. Each primary member of the library was generated by assembling the identically named member of a source library consisting of statements written in AP-101 Basic Assembler Language (BAL). Some source library members produce more than one entry point, in which case load module library ALIAS names are generated for each entry. A macro library was used to standardize frequently used sequences of source code.

## 5.2.2 Purpose

The run time library is used to supply routines, data and interfaces which are needed to execute a HAL/S program or group of programs, which are not produced by the compiler's code generator. Most of the library consists of subroutines which are called from compiler generated code in a HAL statement.

## 5.2.3 Intrinsics and Procedure Routines

The library routines are divided into two groups: intrinsics and procedures. The main distinction is that procedure routines save the passed contents of all fixed point registers, while intrinsics do not. For this reason, a procedure can call another routine (e.g. vector (VV10S3) magnitude calls SQRT), but an intrinsic cannot. Intrinsics do not have a new stack level and therefore do not have any stack work areas. Because intrinsics do not save all passed contents of fixed point registers, they cannot restore them, and must not destroy any register contents that must be returned to the calling program. Expansions of the macros within intrinsics routines are different from the expansions within procedure routines.

# 5.2.4 Register Conventions in Run Time Library Routines

## 5.2.4.1 General Purpose Registers R0-R7.

free use; R1-R3, R5-R7:

> return address during calling R4

and exiting intrinsics, otherwise

free use;

stack base; R<sub>0</sub>

Intrinsics: any or all of Rl, R2, R3, R5, R6, R7 can be used Parameters

for parameter passing.

Procedures: any or all of R2, R4, R5, R6, R7 can be used for

parameter passing.

## 5.2.4.2 Floating Registers F0-F6.

free use; F0-F4

may be used only if saved and F6 (F7)

restored at entry and exit;

depending on the individual routine, Parameters

any or all of F0-F4 can be used for

parameter passing.

Only F6 is guaranteed constant across procedure calls.

## 5.2.4.3 Interface Conventions.

In addition to the parameter passing conventions summarized in general form in the previous two sections and given in detail in the individual library routine descriptions, the compiler has information defining the linkage conventions and register usage for each routine. This section contains that information in a list formatted in four columns as follows:

NAME

The primary or secondary entry

point name.

CALL TYPE

Either PROCEDURE or INTRINSIC to distinguish between routines which must be called via the SCAL instruction and those that must be called

using BAL.

BANK<sub>0</sub>

YES indicates that the routine will always reside in Sector 0 of the GPC and may therefore always be called directly (no ZCON needed). NO indicates that the routine may reside in a sector other than 0 and must therefore be called via a long indirect address constant (ZCON).

Registers assumed to be modified

A list of registers which the compiler assumes to be modified across a call to the routine. Any registers not listed may be assumed to remain unmodified and therefore to maintain their previous contents.

Any modifications to compiler or library should be made carefully so as to maintain this interface properly.

NAME	CALL TYPE	BANKO	REGISTERS ASSUMED TO BE MODIFIED
ACCS	PROCEDURE	NC	FO, F1, F2, F3, F4, F5
ASIN	PROCEDURE	NO	FO,F1,F2,F3,F4,F5
ACCSH	PROCEDURE	NC	FO, F1, F2, F3, F4, F5
ASINH	PROCEDURE	NC	F0,F1,F2,F3,F4,F5
ATANH	PROCEDURE	NO	FO, F1, F2, F3, F4, F5
BTOC	INTRINSIC	NO,	R1, B2, R3, R4, R5, R6, R7
CASPV	INTRINSIC	NO	R1, R2, R3, R4, R5, R6
CASP	INTRINSIC	NO	R1,R2,R3,R4,R5,R6
CASRPV	PROCEDURE	NO	NONE
CASRP	PROCEDURE	NO	ANONE CONTRACTOR CONTRACTOR CONTRACTOR
CASRV	FROCEDURE	NC	NONE
CASR	PROCEDURE	NO	NONE
CASV	INTRINSIC	NO	R1, R2, R3, R4, R5
CAS	INTRINSIC	ИО	R1,R2,R3,R4,R5
CATV	INTRINSIC	NO	R1, R2, R3, R4, R5, R6, R7, F0, F1
CAT	INTAINSIC	NO	R1,R2,R3,R4,R5,R6,R7,F0,F1
CIN	FROCEDURE	NO	LE NONE PROPERTY OF THE WAY OF THE WAY
CINDEX	PROCEDURE	ИС	R5,F0,F1,F2,F3,F4,F5
CINP	PROCEDURE	ИО	
CLJSTV	PROCEDURE	NC	
COUTP	FROCEDURE	NO	
COUT	PROCEDURE	NO	NONE
CPAS	PROCEDURE	NO	
CPASP	PROCEDURE	NO	
CPASE	PROCEDURE	NO	file FO, F1
CPASRP	PROCEDURE	NO	
CPB	INTHINSIC	ИО	R2, R3, R4, R5, R6
CPRC	INTRINSIC	NO	R2, R3, R4, R5, R6
CPRA	PROCEDURE	NO	HOLLINONE TO THE THE SECTION OF THE THE THE THE THE THE TOTAL THE
CRJSTV	PROCEDURE	NC	FO, F1
CSHAPQ	PROCEDURE	ИС	FO, F1, F2, F3, F4, F5
CSLD	PROCEDURE	NO	The R5,F0,F1 and the second of the second of the
CSLDP	PROCEDURE	NO	R5, F0, F1
CPSLD	PROCEDURE	NO	1 . R5, F0, F1
CPSLDE	PROCEDURE	NO	R5, F0, F1
CSST	PROCEDURE	NO	E. S. R. S. F. F. D. F. T. S.
CSSTP	PROCEDURE	NO	R5,F0,F1
CPSSI	PROCEDURE	NO	I R5, F0, F1 ( FL LIBE, B) STORES
CPSSTF	PROCEDURE	NO	A - R5, F0, F1
CSTR	PROCEDURE	NO	THE NONE AND THE LOSS OF THE PARTY OF THE PARTY OF THE PARTY.
CSTRUC	INTRINSIC	NO	R2, R3, R4, R5, R6
CICB	PROCEDURE	NC	R5, P0, F1
CTCE	PROCEDURE	ИО	FO, F1, F2, F3, F4, F5
CTCD	PROCEDURE	NO	FO, F1, F2, F3, F4, F5
CICI	PROCEDURE	NO	TO R5, F0, F1
CTCK	PROCEDURE	NO	R5,F0,F1
CTCH	PROCEDURE	NO	R5,F0,F1
CTCX	PROCEDURE	NO	Transfer Fo, F1 (1) 1 Pt drive have been proceedings
CTCO	PROCEDURE	NO	
CTRIMV	PROCEDURE	ИС	: : : E9, F1 [ ] - [ : [ [ ] ] [ : [ : ] ] - [ : [ : ] ] - [ : [ : ] ] - [ : [ : ] ] - [ : [ : ] ] - [ : [ : ]
DACOS	PROCEDURE	NO	F0, F1, F2, F3, F4, F5
DASIN	PROCEDURE	NO	FO,F1,F2,F3,F4,F5
DACOSH	PROCEDURE	NO	FO, F1, F2, F3, F4, F5

DASINH	PROCEDURE	NC	F0,F1,F2,F3,F4,F5
DATANH	PROCEDURE	NC	F0,F1,F2,F3,F4,F5
DATAN2	PROCEDURE	NO	FO, F1, F2, F3, F4, F5
DATAN	PROCEDURE	NC	F0,F1,F2,F3,F4,F5
DEXP	PROCEDURE	NO	F0,F1,F2,F3
DLCG	PROCEDURE	NO	F0,F1,F2,F3,F4,F5
DMAX	INTRINSIC	NC	R2, R4, R5, F0, F1
DMCVAL	PROCEDURE	NC	F0,F1,F2,F3,F4,F5
DMIN	INTRINSIC	NC	R2,R4,R5,F0,F1
DMOD	INTRINSIC	NC	R4,F0,F1,F2,F3,F4,F5
LPROL	INTRINSIC	NO	R2,R4,R5,F0,F1
DPWHD	FROCEDURE	NC	F0,F1,F2,F3,F4,F5
DPWRI	PROCEDURE	NO	F0, F1, F2, F3
DPWRH	PROCEDURE	NO	F0,F1,F2,F3
DSIN	PROCEDURE	NO	F0,F1,F2,F3,F4,F5
DCCS	PROCEDURE	NC	FO F1 F2 F3 F4 F5
DSINH	PROCECURE	NC	F0,F1,F2,F3,F4,F5
DCCSH	PROCEDURE	NO	F0, F1, F2, F3, F4, F5
DSLD	PROCEDURE	NO	F0,F1,F2,F3,F4,F5
DSQRT	PROCEDURE	NO	
DSST	PROCEDURE	NO	FO,F1,F2,F3,F4,F5
DSUM	INTRINSIC	NC	
DTAN	PROCEDURE	NC	R2,R4,R5,F0,F1
DTANH	PROCEDURE	NC	F0, F1, F2, F3, F4, F5
EATAN2	PROCEDURE	NO	F0,F1,F2,F3,F4,F5
ATAN	PROCEDURE	NC	F0, F1, F2, F3, F4, F5
EMAX	INTRINSIC	NO	F0,F1,F2,F3,F4,F5
EMIN	INTRINSIC	NO	R2,R4,R5,F0,F1
EMCD	INTRINSIC	NO NO	R2,R4,R5,F0,F1
EPROD	INTRINSIC	NC	R4,F0,F1,F2,F3,F4,F5
EPWRE	PROCEDURE	NO	R2,R4,R5,F0,F1
EPWRI	PROCEDURE	NO	F0, F1, F2, F3, F4, F5
EPWRH	PROCEDURE	NC	F0, F1, F2, F3
ESUM	INTRINSIC	NO	F0, F1, F2, F3
EICC	PROCEDURE	ЙС	R2,R4,R5,F0,F1
DTCC	PROCEDURE	NC	F0, F1, F2, F3, F4, F5
EICH	INTRINSIC	YES	FO, F1, F2, F3, F4, F5
DTOH	INTRINSIC	YES	R4, R5, F0, F1
EXP	PROCEDURE	NO	R4, R5, F0, F1
GIBYTE	INTRINSIC	N C	F0, F1, F2, F3
HIN	PROCECURE	NC	R2, R4, R5, F0, F1
IIN	PROCEDURE	NO	F0 F1
EIN	PROCEDURE	No	F0, F1
DIN	PROCEDURE	NC	F.O., F.1
BIN	PROCEDURE	NC	FO, F1
HMAX	INTRINSIC	NO	1.50 FO.F1
HNIN	INTRINSIC	NC	R2,R4,R5,R6
HEROL	INTRINSIC	NO	R2,R4,R5,R6
HSOM	INTRINSIC	NC	R2,R4,R5,R6
IFAX	INTRINSIC	NO	R2, R4, R5, R6
IMIN	INTRINSIC	NC NC	R2,R4,R5,R6
IMOD	INTRINSIC	NO	R2,R4,R5,R6
HMOD	INTRINSIC		R2,R4,R5,R6,R7
ICINIT	PROCEDURE	NO NC	R2,R4,R5,R6,R7
HCUT	PROCEDURE	NC NO	
	CTDOVE	NO	FO, F1

```
REPRODUCIBILITY OF THE
                                                      ORIGINAL PAGE IS POOR
OUTER 1
           PROCEDURE
                             NO
                                           FO.F1
           PROCEDURE
TOUT
                             NO
                                           F0,F1
ECUT
           PROCEDURE
                              NO
                                           F0, F1
DOUT
           PROCEDURE
                             NO
                                           F0,F1
BOUT
           PROCEDURE
                             NO
                                           FO, F1
SKIP
           PROCEDURE
                             NO
                                           FO, F1
LINE
           PROCEDURE
                             NO
                                           F0, F1
COLUMN
           PROCEDURE
                             NO
                                           F0,F1
TAB
           PROCEDURE
                             NC
                                           FO, F1
                             NC
PAGE
           PROCEDURE
                                           FO, F1
IFROD
           INTRINSIC
                             NO
                                           R2, R4, R5, R6, R7
IPWRI
           PROCEDURE
                              NC
                                           R5
           PROCEDURE
IPWRH
                             NO
                                           R5
HPWRH
           PROCEDURE
                             NO
IREM
           INTRINSIC
                             NO
                                           R2,R4,R5,R6,R7
HREM
           INTRINSIC
                             NO
                                           R2.R4.R5.R6.R7
ISUM
           INTRINSIC
                             NO
                                           R2, R4, R5, R6
ITOC
           PROCEDURE
                             NO
                                           NONE
HTCC
           FROCEDURE
                             NO
                                           NONE
ITCD
           INTRINSIC
                             YES
                                           R4, R5, F0, F1
ITCE
           INTRINSIC
                             Y ES
                                           R4, R5, F0, F1
KTQC
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1
LCG
           PROCEDURE
                             NC
                                           FO, F1, F2, F3, F4, F5
           PROCEDURE
MERDNE
                             NC
                                           NONE
MERSNE
           PROCEDURE
                             NO
                                           NONE
MENDNP
           PROCEDURE
                             NO
                                           FO, F1
MHWSNE
           PROCEDURE
                             NC
                                           FO, F1
MMODNE
           INTRINSIC
                             NC
                                           R1, R3, R4, R5, R6, R7, F0, F1
MMOSNP
           INTRINSIC
                             NO
                                           R1, R3, R4, R5, R6, R7, F0, F1
MM1DNP
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3
MM1SNP
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1
MMITNP
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3
MMIWNP
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, FU, F1, F2, F3
MM11DN
           INTRINSIC
                             NO
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3
MM11D3
           INTRINSIC
                             NC
                                           R1, R2, R4, R5, F0, F1, F2, F3, F4, F5
MM11SN
           INTRINSIC
                             NC
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1
MM11S3
                             NC
           INTRINSIC
                                           R1, R2, R4, R5, F0, F1, F2, F3
MM12DN
           PROCEDURE
                             NC
                                           F0,F1,F2,F3,F4,F5
MM12D3
           PROCECURE
                             NC
                                           FO, F1, F2, F3, F4, F5
MM12SN
           PROCEDURE
                             NC
                                          F0,F1,F2,F3,F4,F5
MM12S3
           PROCEDURE
                             NC
                                           FO, F1, F2, F3, F4, F5
                                           R2,R4,R5,R6,F0,F1
MM 13 DN
           INTRINSIC
                             NC
MM13D3
           INTRINSIC
                             NO
                                           R2, R4, F0, F1
           INTRINSIC
MM13SN
                             NO
                                          R2.R4.R5.R6.F0.F1
MM1353
           INTRINSIC
                             NO
                                          R2, R4, F0, F1
MM14DN
           PROCEDURE
                             NC
                                          FO,F1,F2,F3,F4,F5
MM14D3
           PROCEDURE
                             NO
                                           FO, F1, F2, F3, F4, F5
MM14SN
           PROCEDURE
                             NC
                                          FO,F1,F2,F3,F4,F5
MM1453
           PROCEDURE
                             NO
                                          F0, F1, F2, F3, F4, F5
MM15DN
           INTRINSIC
                             NO
                                          R1, R4, R5, K6, K7, F0, F1, F2, F3
MM15SN
           INTRINSIC
                             NO
                                          R1, R4, R5, R6, R7, F0, F1, F2, F3
MM17D3
           PROCEDURE
                             NC
                                          FO,F1,F2,F3,F4,F5
MM17DN
           PROCEDURE
                             NO
                                          FO, F1, F2, F3, F4, F5
MM1753
           PROCEDURE
                             NC
                                          FO, F1, F2, F3, F4, F5
MM 17 S N
           PROCEDURE
                             NC
                                          FO, F1, F2, F3, F4, F5
```

```
R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
MM6DN
           INTRINSIC
                             NO
                                          R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                             NO
MM6D3
           INTRINSIC
                                          R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                             NO
MM6S N
           INTRINSIC
                                          R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                             NO
           INTRINSIC
MM6S3
           PROCEDURE
                                           FO.F1
                             NO
MRODNE
                                          FO,F1
                             NO
MROSNP
           PROCEDURE
                                          F0,F1
           PROCEDURE
                             NC
MR 1DNP
                                          FO,F1
                             NC
           PROCEDURE
MR 1SNP
                             NC
                                           F0,F1
           PROCEDURE
ER 1T NE
                                           FO.F1
                             NC
MRIWNP
           PROCEDURE
                             NO
                                           NONE
MSTR
           PROCELURE
                             NC
                                           R1, R2, R4, R5, R6
           INTRINSIC
MSTRUC
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                             NO
MV6DN
           INTRINSIC
                                           R1.R2,R3,R4,R5,R6,R7,F0,F1,F2,F3
                             NO
MV6D3
           INTRINSIC
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
           INTRINSIC
                             NO
EV6SN
                                           R1, R2, R3, R4, R5, R6, F0, F1, F2, F3
                             NO
MV6S3
           INTRINSIC
                             NC
                                           FO, F1
           PROCEDURE
QSHAPQ
                                           F0,F1,F2,F3
                             NO
           PROCEDURE
BANDOM
                                           FO.F1.F2.F3
                             NO
RANDG
           PROCEDURE
                             YES
                                           K4,R5,F0,F1
           INTRINSIC
ROUND
                                           R4, R5, F0, F1
                             Y ES
ETOI
           INTRINSIC
                                           R4, R5, F0, F1
           INTRINSIC
                              YES
TRUNC
                                           R4, R5, F0, F1
           INTRINSIC
                              YES
FLOOR
                                           R4, R5, F0, F1
                              YES
CEIL
           INTRINSIC
                                           R4, R5, F0, F1
                             YES
DIRUNC
           INTRINSIC
                                           R4,R5,F0,F1
           INTRINSIC
                              YES
DFLOCE
           INTRINSIC
                                           R4,R5,F0,F1
                              YES
DCELL
                                           R4, R5, F0, F1
                              YES
DROUND
           INTRINSIC
                                           R4, R5, F0, F1
                              YES
DICI
           INTRINSIC
                                           R1,R3,R4,F0,F1,F2,F3,F4,F5
                              NC
SIN
           INTHINSIC
                                           R1, R3, R4, F0, F1, F2, F3, F4, F5
           INTRINSIC
                              NC
CCS
                                           FO,F1,F2,F3,F4,F5
           PROCEDURE
                              NC
SINH
                                           FO, F1, F2, F3, F4, F5
                              NC
CCSH
           PROCEDURE
                                           R1, R4, R5, R6, R7, F0, F1, F2, F3
           INTRINSIC
                              NO
SORT
                                           R1, R4, R5, F0, F1
                              NC
           INTRINSIC
SIBYIE
                                           FO,F1,F2,F3,F4,F5
                              NC
           PROCEDURE
TAN
           FROCEDURE
                              NC
                                           F0,F1,F2,F3,F4,F5
TANH
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                              NO
V M 6D N
           INTRINSIC
                                           R1, R2, R3, R4, R5, F0, F1, F2, F3, F4, F5
                              NG
VM6D3
           INTRINSIC
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5
                              NO
VM65N
           INTRINSIC
                                           R1, R2, R3, R4, R5, F0, F1, F2, F3
                              NO
VM6S3
           INTRINSIC
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F4, F5
                              NO
VC6DN
           INTRINSIC
                                           R1, R2, R3, R4, R5, R6, F0, F1
                              NC
            INTRINSIC
VC6D3
                                           R1, R2, R3, R4, R5, R6, R7, F0, F1, F4, F5
                              NO
V065 N
           INTRINSIC
                                           R1, R2, R3, R4, R5, R6, F0, F1
                              NO
           INTRINSIC
V0653
                                           FO, F1
                              NO
            PROCEDURE
VRODN
           PROCEDURE
                              NC
                                           FO, F1
VRODNE
                                           FO, F1
                              NC
VROSN
            PROCEDURE
                                           FO.F1
                              NC
            PROCEDURE
VROSNE
                                           FO,F1
                              NO
           FROCEDURE
VR 1DN
                                           F0, F1
            PROCEDURE
                              NO
VE 1DNE
            PROCEDURE
                                           FO, F1
                              NC
VR1SN
                                           F0, F1
                              NC
VEISNE
            PROCEDURE
                                           FO,F1
                              NC
            PROCEDURE
VR 1T N
                              NO
                                           FO,F1
            PROCEDURE
VRITNE
```

```
FO, F1
           PROCEDURE
                              NC
VR1WN
                                           FO,F1
           PROCEDURE
                             NC
VR 1WNF
                                           R1, R4, R5, F0, F1
           INTRINSIC
                              NC
VVODN
                                           R1,R4,R5,R7,F0,F1
                              NC
VVODNP
           INTRINSIC
                                           R1, R4, R5, F0, F1
VVOSN
           INTRINSIC
                              NC
                                           R1,R4,R5,R7,F0,F1
           INTRINSIC
                              NC
VVOSNE
                                           R1, R2, R4, R5, F0, F1
VV1DN
           INTRINSIC
                              NС
                                           R1, R2, R4, F0, F1, F2, F3, F4, F5
                              NC
           INTRINSIC
V V 1D 3
                                           R1, R2, R4, R5, R6, R7, F0, F1
                              NO
VV1D3P
           INTRINSIC
                                            R1, R2, R4, R5, R6, R7, F0, F1
VV 1DNP
           INTRINSIC
                              NC
                                            R1, R2, R4, R5, F0, F1
                              NC
VV1SN
           INTRINSIC
                                            R1, R2, R4, F0, F1, F2, F3, F4, F5
                              NC
VV153
           INTRINSIC
                                            R1, R2, R4, R5, R6, R7, F0, F1
                              NO
VV1S3P
           INTRINSIC
                                            R1, R2, R4, R5, R6, R7, F0, F1
                              NO
VV1SNP
           INTRINSIC
                                            R1, R2, R4, R5, F0, F1
                              NC
VV1TN
           INTRINSIC
                              NC
                                            R1, R2, R4, F0, F1, F2, F3, F4, F5
V V 1T3
           INTRINSIC
                                            R1, R2, R4, R5, R6, R7, F0, F1
           INTRINSIC
                              NO
VV1T3P
                                            R1, R2, R4, R5, R6, R7, F0, F1
                              NO
VVITNE
           INTRINSIC
                                            R1, R2, R4, R5, F0, F1
                              NO
VVIHN
           INTRINSIC
                              NO
                                            R1, R2, R4, F0, F1
           INTRINSIC
VV1W3
                                            R1, R2, R4, R5, R6, R7, F0, F1
                              NO
VV1W3P
           INTRINSIC
                                            R1, R2, R4, R5, R6, R7, F0, F1
                              ΝO
VVIWNP
           INTRINSIC
                                            FO, F1, F2, F3, F4, F5
                              NC
VV10D3
           PROCEDURE
                                            FO,F1,F2,F3,F4,F5
                              N C
VV10DN
           PROCEDURE
                                            FO, F1, F2, F3, F4, F5
                              NO.
VV9D3
           FROCEDURE
                                            FO,F1,F2,F3,F4,F5
                              NC
VV9DN
           PROCEDURE
                                            FO.F1, F2, F3, F4, F5
                              NC
           PROCEDURE
VV1053
                                            FO,F1,F2,F3,F4,F5
                              NC
VVIOSN
           PROCEDURE
                                            FO, F1, F2, F3, F4, F5
                              NC
VV953
           PROCEDURE
                                            F0,F1,F2,F3,F4,F5
                              NC
           PROCEDURE
VV9SN
                                            R1, R2, R3, R4, R5, F0, F1
                              NO
           INTRINSIC
VV2DN
                                            H1, R2, R3, R4, F0, F1, F2, F3, F4, F5
                              NC
            INTRINSIC
VV2D3
                                            R1, R2, R3, R4, R5, F0, F1
                              NO
VV2SN
            INTRINSIC
                                            R1, R2, R3, R4, F0, F1, F2, F3, F4, F5
                              NC
            INTRINSIC
VY2S3
                                            R1, R2, R3, R4, R5, F0, F1
                              NO
VV3DN
            INTRINSIC
                                            R1, R2, R3, R4, F0, F1
                               NC
V V 3D3
            INTRINSIC
                                            R1, R2, R3, R4, R5, F0, F1
                               NO
VV3SN
            INTRINSIC
                                            R1, R2, R3, R4, F0, F1, F2, F3, F4, F5
                               NC
VV3S3
            INTRINSIC
                               N C
                                            R1, R2, R4, R5, F0, F1, F2, F3
            INTRINSIC
VV4DN
                                            R1, B2, R4, F0, F1, F2, F3
                               NC
VV4D3
            INTHINSIC
                                            R1, R2, R4, R5, F0, F1, F2, F3
                               NC
            INTRINSIC
VV4SN
                                            R1, H2, R4, F0, F1, F2, F3
                               NO
VV453
            INTRINSIC
                                            R1, R2, R4, R5, F0, F1, F2, F3
VV5DN
            INTRINSIC
                               NO
                                            R1, R2, R4, F0, F1, F2, F3, F4, F5
                               NC
 VV5D3
            INTRINSIC
                                            R1, R2, R4, R5, F0, F1, F2, F3
                               NO
            INTRINSIC
 V V 55 N
                                            R1, R2, R4, F0, F1, F2, F3
                               NC
 V V 55 3
            INTRINSIC
                                            R1, R2, R3, R4, R5, F0, F1, F2, F3
                               NO
 VV6DN
            INTRINSIC
                                            R2, R3, R4, F0, F1, F2, F3
                               NO
 VV6D3
            INTRINSIC
                                            R1, R2, R3, R4, R5, F0, F1, F2, F3
                               NO
            INTRINSIC
 VV65N
                                            R2, R3, R4, F0, F1, F2, F3
                               NC
 VV653
            INTRINSIC
                                            R1, R2, R4, R5, F0, F1
                               NC
 VV7DN
            INTRINSIC
                                            R1, R2, R4, F0, F1, F2, F3, F4, F5
                               NC
            INTRINSIC
 V V 7D 3
                                             R1, R2, R4, R5, F0, F1
                               NC.
            INTRINSIC
 VV7SN
                                             R1, R2, R4, F0, F1, F2, F3, F4, F5
                               NC
 V V7S3
            INTRINSIC
                                             R1, R2, R3, R4, R5, F0, F1
            INTRINSIC
                               NO
 VV8D3
```

VVBDN	INTRINSIC	NC	R1, R2, R3, R4, R5, F0, F1
V V 85 3	INTRINSIC	NO	R1, B2, R3, R4, R5, F0, F1
VV8SN	INTRINSIC	NO	R1, R2, R3, R4, R5, F0, F1
V X 6 D 3	INTRINSIC	NC	R1, R2, R3, R4, F0, F1, F2, F3, F4, F5
V X 65 3	INTRINSIC	NO	R1,R2,R3,R4,F0,F1,F2,F3
XTOC	INTRINSIC	NO	R1, R2, R3, R4, R5, R6, R7, F0, F1
OTOC	INTRINSIC	NO	R1, R2, R3, R4, R5, R6, R7, F0, F1

# 5.2.5 Referencing Conventions

In order to comply with the CSECT 5.2.5.1 CSECT Names. naming standards described in the HAL/SDL ICD, all library code CSECTs begin with two alphabetic characters (A-Z)\*. All library primary names and aliases are unique to 6 characters.

Whenever a data CSECT is needed for a particular library module, it is given the CSECT name #Lnnnnnn, where nnnnnn is the first 6 characters of the primary entry name.

5.2.5.2 ZCON's. For each primary entry point and alternate entry point in the runtime library, a member exists in a separate ZCON library. The members in the ZCON library contain address constants which refer to the actual entry points. Thus, for the library routine named SIN which has an entry point named COS, there are two members in the ZCON library named #QSIN and These #Q modules contain references to the respective entry points. The individual ZCONs in the ZCON library are created by assembly code like the following:

> **#OSIN CSECT** DC Z(SIN,,X'E') EXTRN SIN END

Some library routines make reference to other library routines via the ACALL macro (see Section 5.2.6). The ACALL macro does not make reference via a ZCON as is done when compiler-emitted code references a library routine. Use of the ACALL macro for inter-library routine reference means that a referenced routine must be in the same machine sector as the referencing routine, or must be in sector zero.

Sector 0 routines are an exception: their CSECT names begin with #0. This is to conform to link editor conventions for routines which must reside in sector 0. Sector 0 routines are identified in the list in Section 5.2.4.3 and in the boxed area of the individual library routine description.

## 5.2.6 Coding Structure

The following outline represents the standard coding structure of all library members.

- 1 TITLE
- 2 WORKAREA macro definition used only if addition stack storage was needed
- 3 AMAIN
- 4 \* Comment card describing the function of the primary entry point
- 5 INPUT
- 6 OUTPUT
- body of executable code including use of WORK, AERROR, AEXIT macros where needed and alternate entry points defined using the AENTRY macro, function comment card, and INPUT and OUTPUT macros in the same manner as the primary entry point.
- 8 DC constant area addressed via PC relative mode
- 9 ADATA, followed by a DC constant area addressed via base and displacement mode.

used only if constands need to be indexed

10 ACLOSE

## 5.2.7 The Macro Library

To standardize interface conventions, automate production of commonly used code sequences, and impose a structure to the runtime library, a series of macros are used. This section describes the function, use, and expansion of these macros. Lower case letters are used to indicate variable fields. Square brackets [] indicate optional fields, braces {} indicate a choice of required fields. Complete listings of all the macro source code is also included.

#### AMAIN

11

name AMAIN 
$$\begin{bmatrix} INTSIC = \begin{cases} YES \\ INTERNAL \end{cases} \\ ACALL = YES \\ SECTOR = 0 \end{bmatrix}$$

## Function:

Defines "name" as the primary entry point of a routine.

#### INTSIC=YES:

Defines the routine (and any entry points) as an intrinsic. If the INTSIC operand is omitted, the routine is defined as a procedure.

#### INTSIC=INTERNAL:

Defines an intrinsic which is called only by other routines in the library. At present, this is only GTBYTE and STBYTE.

#### ACALL=YES:

(Valid only for procedure routines.) Allows use of the ACALL macro within the routine (See ACALL description).

#### SECTOR=0:

Defines the routine (intrinsic or procedure) as a Sector 0 routine.

#### Expansion:

The macro first defines the primary entry "name" (the AMAIN label) as the CSECT name, unless SECTOR=0 was specified. In the latter case, the CSECT name is generated by prefixing "name" with #0, and the primary entry "name" is defined using the DS and ENTRY statements. The options selected via the AMAIN operands are saved in global SETB variables for testing by the other macros. If either INTSIC option was selected, the macro ends. Otherwise, a procedure is being defined, so the STACK DSECT is generated.

The STACK DSECT consists of a standard 18 halfword area. including symbols for the saved copies of the fixed point register parameters (ARG2, ARG4, ARG5, ARG6, ARG7), followed by the WORKAREA macro. The WORKAREA macro is the means by which additional storage beyond the standard stack of 18 halfwords may be defined. If such storage is needed a local WORKAREA macro must have been defined earlier in the source which contains the appropriate DS assembler statements. statements are thus incorporated as the remainder of the If additional storage is not needed, the local STACK DSECT. WORKAREA macro is not defined. As a result, the system WORKAREA macro is invoked, which does not define any storage, leaving the STACK DSECT at its standard length. The system WORKAREA macro also sets a global SETB variable, which is tested later by the AMAIN macro to determine if the stack is standard or augmented. The STACK DSECT is then terminated by resuming the The STACK DSECT is defined in this sequence original CSECT. so that the assembler will output the SYM records in the order expected by the link editor's stack size algorithm. A USING statement is generated to give addressibility to the stack area. Finally, the executable code of the entry prologue is generated. This consists of an NIST instruction to zero the 9th halfword of the new stack frame, establishing a null ON ERROR environment. In addition, if both ACALL=YES is specified and a local WORKAREA provided, the default stack size of 18 set up by the SCAL microcode will be insufficient, so an IAL to set up the new stack size is generated.

#### AENTRY

name AENTRY

#### Function:

Defines "name" as a secondary entry point.

#### Expansion:

"name" is externally defined using the DS and ENTRY statements. If the routine was defined as an intrinsic, the macro ends. Otherwise, the executable code of the entry prologue is generated in the same manner as the AMAIN macro.

#### AEXIT

AEXIT 
$$\begin{bmatrix} CC = \begin{pmatrix} KEEP \\ (rx) \\ EQ \\ NE \\ COND=code \end{bmatrix}$$

#### Function:

Cause return of control from a procedure or intrinsic routine.

CC:

Used to pass a condition code back to the caller. It can be used only if OUTPUT CC was specified. (See OUTPUT macro.)

## Valid for Intrinsics Only:

#### CC=KEEP:

Passes back the condition code as is.

#### CC=(rx):

Passes back the condition code generated by a LR rx, rx.

## Valid for Procedures Only:

#### CC=EQ:

Passes back an equal (B'00') condition code.

#### CC=NE:

Passes back a not equal (B'll') condition code.

Note: The CC= operand is used in the following 8 routines: CPR, CPRA, CTSR, CSTRUCT, VV8DN, VV8D3, VV8SN, and VV8S3.

#### COND=code:

Used to do a conditional return, i.e. based on the current condition code. Valid for procedures only. "code" is either a number used as the mask on a BC opcode, or a letter or letter pair representing the mask in the extended BC mnemonic op codes. (E, Z, NE, NZ, H, O, L, M, HE, LE, NL, NM, NH, NO). This operand may be used to improve the efficiency of some routines. If used, be sure valid executable code follows it, so the fall through case is valid.

## Expansion:

The code generated by the AEXIT macro depends primarily on whether the routine is an intrinsic or procedure, and secondarily on what operands were supplied, and, in the case of intrinsics, what fixed point registers were used. The expansions for intrinsics and procedures are described separately.

## Intrinsics:

If register(s) Rl and/or R3 have been defined (see INPUT, OUTPUT, and WORK macros), it is assumed they have been modified and must be restored from the stack, since they are the addressing registers for compiled code. This is done via the appropriate LH instruction(s), or IHL and SLL instructions if CC=KEEP was specified, since LH would destroy the existing condition code. If CC=(rx) was specified, a LR rx,rx is generated to set the condition code. Finally, a BCRE or BCR is generated to cause a return to the caller. A BCR is generated if SECTOR=0 or INTSIC=INTERNAL was specified on the AMAIN macro.

#### Procedures:

If CC=EQ or CC=NE was specified, the condition code bits in the return PSW in the stack are zeroed or set via the ZB or SB instruction. Then, an SRET instruction is generated with a mask of 7 if the COND operand was omitted, or the appropriate mask if it was supplied.

#### INPUT

INPUT {register spec type comments}

#### Function:

Defines input interface of primary or alternate entry point and symbolic names for the register(s).

## Register Spec:

One of R1, R2, R3, R4, R5, R6, R7, F0, F1, F2, F3, F4, F5, F6, or F7. If there is no input (RANDOM, RANG only), code NONE. If there is more than one, use continuation lines for each subsequent one (see examples).

#### Type Comments:

type	precision	units
SCALAR	SINGLE/DOUBLE	RADIANS
MATRIX (3,3)		
MATRIX (N, N)		
VECTOR (3)		
VECTOR (N)		
INTEGER (N)		
CHARACTER		

#### Examples:

		col. 16				col. 72
(1)	INPUT	F0	SCALAR	SINGLE	RADIANS	
(2)	INPUT	R2,		DOUBLE		<b>X</b>
		R3, R5	VECTOR (N) INTEGER (N)	DOUBLE SINGLE		X

Note: R1 and R3 are illegal inputs for procedure routines, and R4 is illegal for intrinsic routines.

## Expansion:

For each register spec supplied, the macro checks for a valid register symbolic, or for the special case of NONE. If the symbolic register name has not been previously defined, an EQU statement is generated to define it. The macro also tests for the illegal use of Rl or R3 for a procedure parameter and R4 for an intrinsic. A global arrayed SETB variable is set, which in conjunction with the AMAIN, AENTRY, and ACLOSE macros, will guarantee that an INPUT macro has been supplied for each entry point (see ACLOSE macro).

#### OUTPUT

OUTPUT { register spec type comments NONE CC

#### Function:

Defines output interface of primary or alternate entry point.

Operand form is identical to that of INPUT macro, with the addition of CC as a possibility. This indicates that the condition code is the output of the routine. If CC is specified, the CC= option of the AEXIT macro must be used.

#### Expansion:

Same as for INPUT macro, except for special processing for the CC operand. If CC is supplied, a global SETB variable is set which is tested by the AEXIT macro for consistency with its CC operand.

#### WORK

WORK {register spec}

#### Function:

Defines work registers.

## Expansion:

Similar to INPUT and OUTPUT, except that this macro is required only if additional register symbols need to be defined.

#### ABAL

ABAL name

## Function:

Calls the intrinsic routine "name", valid only in a procedure routine.

## Expansion:

Generates a BAL 4, name, and an EXTRN statement if "name" has not been previously defined.

#### ACALL

ACALL name

#### Function:

Calls the procedure routine "name", valid only in a procedure routine defined with ACALL=YES option.

#### Expansion:

Generates an SCAL 0, name, and an EXTRN statement if "name" has not been previously defined.

#### AERROR

AERROR number cause comment

#### Function:

Generates a send error SVC instruction to signal a run time error to the FCOS.

#### Number:

The error number.

#### Cause Comment:

Brief description of the cause of the error.

## Expansion:

This macro accumulates, in GBLA variables, all errors sent within one assembly. It also checks to see that the error number indicates as an argument to AERROR is less than a maximum value. The actual code emitted is an SVC in which the operand is the label of an SVC parameter list to be emitted by the ADATA or ACLOSE macro via the ERRPARMS macro. If any error is sent more than once in an assembly, AERROR insures that only one SVC parameter list for that error is used.

#### ADATA

ADATA

#### Function:

Defines the start of a separate data CSECT for indexable constant data.

#### Expansion:

A CSECT is created with the name #Lnnnnnn where nnnnnn is the first 6 characters of the primary CSECT name defined by the AMAIN macro. The ADATA macro ends leaving the data CSECT in effect so that any user-defined data following the macro call will be part of the data CSECT. The ERRPARMS macro is invoked so that any possible AERROR SVC parameter lists will appear before the indexed data. This is necessary so that the assembler will use the direct addressing mode instead of base and displacement.

## ACLOSE

ACLOSE

#### Function:

Terminates the assembly.

## Expansion:

The macro first invokes the ERRPARMS macro to create the AERROR SVC parameter lists. (See ERRPARMS macro.) It then checks via arrayed global SETB variables if INPUT and OUTPUT macros were supplied for each entry point. Finally, it generates an END assembler statement, terminating the assembly.

#### FRRPARMS

ERRPARMS

#### Function:

Generates SVC parameter lists for the AERROR macro.

## Expansion:

This macro is invoked by the ADATA and ACLOSE macro. It first tests a global SETB variable to see if it has already been invoked, in which case the macro does nothing. Otherwise, it generates a CSECT statement to define the data CSECT (FCOS parameter lists must reside in the data sector). The CSECT name is #Lname, where "name" is the primary entry name. The parameter lists are generated by looping through arrayed global SETA variables in which the AERROR macro saved the unique error numbers. ERRPARMS is invoked by the ADATA macro because the parameter lists must be before any indexed data following the optional ADATA macro. It is invoked by the ACLOSE macro in case the ADATA macro is not used.

#### WORKAREA

#### WORKAREA

## Function:

An automatically invoked, user-created macro used to define extensions of the stack area for temporary reentrant storage. The WORKAREA macro is invoked by the AMAIN macro in procedure routines. A system supplied default is invoked in the absence of a user-created macro.

## Expansion:

The system WORKAREA macro merely sets a global SETB variable which is tested by the AMAIN macro to determine whether the system or user macro is being expanded.

# REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

## AMAIN

ENAME  CCSECT  EENTCNT  ENAMES (EE ***********************************	GBLC &CSECT, ENAMES SETC *ENAME* SETA &ENTONT+1 NTCNT) SETC *ENAME* ***********************************	ENOEXTRA, &INTERN, & SECTO (20)  ***********************************	00001000 00001100 00001200
******	*******	*********	00001300
& CNAME	SETC 'ENAME'		00001400
	AIF ( ESECTOR EQ	**) .REG	00001500
	AIF ( * ESECTOR * NE	*O*).BADSECT	00001600
SCNAME	SETC #0 . SNAME		00001700
ESECT 0	SETB 1		00001800
SCNAME	CSECT		00001900
SNAME	DS OH	PRIMARY ENTRY POINT	00002000
	ENTRY ENAME		00002100
	AGO .COMM		00002200
REG	ANOP		00002300
SCNAME	CSECT		00002400
.COMM	ANOP		00002500
ELIB	SETB ("EINTSIC" EQ	s (NO)) is in the second library of programmed the second	00002600
EINTERN	SETH ('BINTSIC' EQ	"INTERNAL")	00002700
	AIF (NOI ELIB) . SP	ACE communication in the first of the communication of	00002800
STACK	DSECT		00002900
*	DS 18H	STANDARD STACK AREA DEFINITION	00003000
	DS F	PSW (LEFT HALF)	00003100
	DS. 2F	120 RO, R1 - File (2000) [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [	00003200
ARG2	CS F	$\mathbb{R}^{2} \otimes \mathbb{R}^{2}$ . The first section is the proof of the set $\mathbb{R}^{2} \otimes \mathbb{R}^{2}$ , which is the $\mathbb{R}^{2} \otimes \mathbb{R}^{2}$	00003300
	DS F	R3 · 1	00003400
ARG4	rs F		00003500
ARG5	DS F		00003600
ARG6	DS F		00003700
ARG7	ES F		00003800
*	LND CF STANDARD STA	ACK AREA	00003900
	WORKAREA		00004000
STACKEND		END OF COMBINED STACK AREA	00 00 4 100
&CNAME	CSECT		00004300
	USING STACK, 0	ADDRESS STACK AREA	00004400
ECALLCK	SETB ( SACALL EQ		00004500
	AIF (ENCEXTRA OR	NOT ECALLOK) . NIST	00004600
1		FACK SET STACK SIZE  CLEAR ON ERROR INFO (LCL DATA PTR)	00004700
NIST	NIST 9 (0),0	CTDWW ON FUNOW INTO (FOR PAIR EIN)	00004700
.SPACE	SPACE	사는 그리는 사는 사람들에 가는 하는 것이 하다는데 이렇게 했다.	00004900
	MEXIT	DR=0 MAY BE SPECIFIED'	00005000
, BADS ECT	MEND		00005100

## AENTRY

& NAME	MACRC AENTRY GBLA SENTONT GBLE SCALLOK, SNOEXTRA, SLIB	00 000 100 00 000 200 00 000 300 00 000 400
	GBLC ENAMES (20)	00000500
******	SECONDARY ENTRY POINT	00000600 00000700 00000800 00000900
EENTONT ENAMES (E	SETA & ENTONT+1 ENTONI) SETC "& NAME"	00001100 00001200
ENAME	DS OH ENTRY ENAME AIF (NOT ELIB). SPACE	00001300 00001400 00001500
	AIF (ENCEXTFA OR NOT ECALLOK) .NIST IAL O, STACKEND-STACK SET STACK SIZE	00001600 00001700 00001800
.NIST .SPACE	NIST 9(0),0 CLEAR ERBOR VECTOR POINTER SPACE MEND	00001900

## AEXIT

	MACRC		00000100
ENAME	AEXIT	&CC=, &COND=	00000200
	GBLA	ERET BEFORE THE SECOND OF THE SECOND SEC	00000300
	GBLB	ELIB, ECCTYPE, EINTERN, ESECTO	00000400
	LCLA	& MASK	00000500
EMASK	SETA		00000600
ERET	SETA	8 x ET + 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1	00000700
*****	RETUR	A LO CYLLEB ***********************	000C0800
ENAME	DS		00000900
	AIF	(*ECC* EQ ** AND NOT ECCTYPE).OK1	00001000
	AIF	(*SCC* NE ** AND SCCTYPE).OK1	00001100
	MNOTE	1, CONFLICTING CC OPERANDS IN OUTPUT AND AEXIT MACROS	00001200
OK1		(CLIB) .LIB	00001300
. *		ATE EXIT SEQUENCE FOR INTRINSICS	00001400
	AIF	('SCOND' EQ '').OK2	00001500
	MNOTE	4, COND OPERAND INVALID FOR INTRINSIC.	00001600
.OK2	AIF	('SCC'LEQ-').LHS (" TENNER DE LE SENTE	00001700
		('SCC' (1,2) EQ '(R').LHS	00001800
	AIF	('ECC' EQ 'KEEP') . THLS	00001900
	MNOTE	4, INVALID CC OPERAND FOR INTRINSIC	00002000
.LHS	AIF	(SINTERN).SKIP1	00002100
	AIF	(NOT D'R3).SKIP3	00002200
	LH	3,9(0) RESTORE LOCAL DATA BASE	00002300
.SKIP3	AIF	(NOT D'R1).SKIP1	00002400
	LH	1,5(0) RESTORE PROGRAM DATA BASE	00002500
.SKIP1	AIF	( SCC EQ ) BCRE	00002600
8 R	SETC	THECC (1) In the first of the first of the control of the first of the	00002700
	AIF	( ER NE RI AND ER NE R3) LROK	00002800
	MNOTE	4, INVALID REGISTER IN CC = OPERAND!	00002900
. LROK	LR	ER, ER SET CONDITION CODE	00003000
	AGO.	I. BCRE	00003100

## AFXIT (continued)

```
00003200
.IHLS
        AIF
             (SINTERN) . BCRE
                                                                 00003300
             (NCT D'R3).SKIPR3
        AIF
             R3,9(0)
        IHL
                          LOAD R3, PRESERVING CC
                                                                 00 C 03 4 0 0
        SLL
             R3, 16
                          POSITION IN UPPER HALFWORD
                                                                 00003500
             (NOT D'R1) . ECRE
                                                                 00003600
.SKIPR3 AIF
                                                                 00003700
                          LOAD R1, PRESERVING CC
             R1,5(0)
        IHL
                          POSITION IN UPPER HALFWORD
                                                                 00003800
             R1.16
        SLL
BCRE
                                                                 00003900
        ANOP
              (SSECTO OR SINTERN) . BCR
                                                                . 00004600
        AIF
SRETERET BCRE 7.4
                    RETURN TO CALLER
                                                                 00004100
SPACE
                                                                 00004300
                                                                 00004400
        MEXIT
       ANCP
BCR
                                                                 00004500
                         RETURN TO CALLER
                                                                 00004600
SRETERET ECA
******************
                                                                 00004800
             SPACE
                                                                 00004900
       MEXTT .
              ('&CC' EQ '' OR '&COND' EQ '') .OK3
                                                                00005000
LIB
        ATF
        MNOTE 4, CC AND COND OPERANDS ARE MUTUALLY EXCLUSIVE
                                                                 00005100
. окз
              ("ECC" EQ "") . NOCC
        AIF
                                                                 00005200
              ('ECC' EQ 'EQ').ZB
                                                                 00005300
        AIF
              ('ECC' EQ 'NE').SB
        ATF
                                                                 00005400
        MNOTE 4, INVALID CC OPERAND FOR PROCEDURE ROUTINE
                                                                 00005500
. ZB
        Z.B
              1(0), X'COOO' SET PSW CC TO OO (EO)
                                                                 00005600
        AGO
              . NCCC
                                                                 00005700
.SB
        SB
              1(0), X COOO SET PSW CC TO 11 (LT (NE))
                                                                 00005800
. NOCC
        AIF
              (T'ECOND NE 'N') . CONDTST
                                                                 00005900
EMASK
        SETA &COND
                                                                 00006000
              .SRET
        AGO
                                                                 00006100
.CONDTST AIF
              (*ECOND* EO **) . SRET
                                                                 00006200
        SETA 1
& MASK
                                                                 00006300
        AIF
              ("SCOND" EQ "H" OR "SCOND" EQ "O") .SRET
                                                                 00006400
EMASK
        SETA
                                                                 00006500
        AIF
              ( * &COND * EQ *L * OR * &COND * EO *M *) . SRET
                                                                 00006600
        SETA 3
& MASK
                                                                 00006700
              ('ECOND' EQ 'NE' OR 'ECOND' EQ 'NZ') . SRET
        AIF
                                                                 00006800
EMASK
        SETA 4
                                                                 00006900
              ('ECOND' EQ 'E' OR 'ECOND' EQ 'Z') .SRET
        ATE
                                                                 00007000
EMASK
        SETA
                                                                 00007100
   AIF ('ECCND' EQ 'HE' OR 'ECOND' EQ 'NL' OR 'ECOND' EQ 'NM') . SRET
                                                                 00007200
& MASK -
       SETA 6
                                                                 00007300
    AIF (*ECCND* EQ *LE OR *ECOND EQ 'NH' OR *ECOND EQ NO:) SRET
                                                                 00007400
EMASK
        SETA 7
                                                                 00007500
        MNOTE 4. INVALID COND OPERAND.
                                                                 00007600
SRET
       ANOP
                                                                 00007700
SRETERET SHET EMASK, O
                           RETURN TO CALLER
                                                                 00007800
SPACE
                                                                 00008000
        MEND
                                                                 00008100
```

# INPUT

The transfer MACRO Committee of the comm	
INPUT &X	00000100
GBLA SENTCHT	00000200
GBLB &INPUT (20), &LIB	00000300
ALF (N'ESYSLIST EQ O) . EMPTY	00 000 400
SINPUT (SENTONI) SETB 1	00000500
AIF ('EX' EQ 'NONE'). SPACE	00000600
EI SETA 1	00000700
ELAST SETA N'ESYSLIST	00000800
.LOOP AIF (K'ESYSLIST(EI) NE 2). BADREG	00000900
ER SETC 'ESYSLIST(EI)'	00001000
AIF ('ER' (1,1) NE 'F' AND 'ER' (1,1) NE 'R'). BADREG	00001100
AIF (*ER* EQ *RO*) . BADREG	00001200
AIF (ELIB AND (*ER * EQ *R1 * OR *ER * EQ *R3 *)) . INVREG1	00001300
AIF (NOT ELIB AND 'ER' EQ 'R4') . INVREG 2	00001400
AIF (D.ER) .NEXT	00001500
EN SETC 'ER' (2,1)	00001600
ER EQU EN	00001700
NEXT ANOP	00001800
SI SETA SI+1	00001900
AIF (&I LE &LAST) . LOOP	00002000
SPACE SPACE	00002100
ing talah mexata ali tabuk bilah sebagai di alam atau atau atau atau atau atau atau at	00002110
	00002200
.BADREG MNOTE 4, · ILLEGAL REGISTER SPECIFICATION - &SYSLIST(&I) ·	00002300
그 그 그 그 그 그 그 그 그 그 그 그 그 그 그 그 그 그 그	00002400
.INVREG1 MNOTE 4, ER INVALID INPUT FOR PROCEDURE ROUTINE.	00002500
	00002600
.INVREG2 MNOTE 4, *R4 INVALID INPUT FOR INTRINSIC*  AGO .NEXT	00002700
	00002800
.EMFTY MNOTE 4, *CPERAND REQUIRED *	00002900
보다 : 이글 ####다 이글 이 인생은 첫 번호를 하고 있다. 이 그렇게 되는 ###################################	00003000

## REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

## OUTPUT

	00 000 100
MACEC	00 0 00 200
CUIPUT EX	00000300
AND A CRNTCNT	00 0 00 4 00
COLD SOUTPHT (20) SCCTYPE, GLID	00000500
AIF (N.ESYSLIST EQ O) . EMPTY	00000600
SETR 1	00000700
EOUTPUT (SENICAL) SELO *NONE*).SPACE	00000800
	00000900
WA CCUCTTCT	00001000
CLASI NE Z) . BAURES	00001100
cpmc (ESYSLIST (61)	00001200
AIF (*ER' EQ CC').CCTYPE	00001300
(1.1) NE F AND OR (1.1)	C00C14C0
AIF (*ER* EQ *RO*) . BADREG	00001500
	00001600
	00001700
	00001800
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]	00001900
	00002000
1 OK I THE RESERVE TO THE SECOND OF THE SECOND SECO	00002100
NEXT ANOP	00002200
OI OI	00002210
	00002300
SPACE SPACE MEXIT PROTETER SPECIFICATION - &SYSLIST (&I)	00002400
	00002500
	00002600
[1] 사이라	00002700
CCTYPE ANOP	00002800
SCCTYPE SETB 1 AGC NEXT FOR PROCEDURE ROUTINE	00002900
THE PROPERTY OF THE PROPERTY O	. 00003000
	00003100
THE THREE TO CHAPITAL LOS TRIVINGS	00003200
1 NEXT	00003300
" compand REOHIRED"	00003400
EMPTY MNOTE 4, CPERKAD EXCOLUTION MEND	
protestical contrations (CAPA) in the contract of the contract	

## WORK

	MACRC	00000100
	TO WORK THE SX TO SEE THE SEE	00000200
	GBLB ELIB, ENCEXTRA	00000300
	AIF ('EX' EQ 'NONE'). SPACE	
εı	SETA 1	00000400
ELAST	SETA N'ESYSLIST	00000500
LOCP		00000600
& B		0000070 <b>0</b>
D A	SETC 'ESYSLIST (EI) '	00000800
	AIF ('ER' (1, 1) NE 'F' AND 'ER' (1, 1) NE 'R') . BADREG	00000900
	AIF (*ER* EQ *RO*), BADREG	00001000
	AIF (*ER* NE *P6*).TESTD	00001100
MNOTE	***** WARNING: F6 MUST BE PRESERVED ACROSS CALLS*	00001200
. TESTD	AIF (D * ER) . NEXT	00001200
εn	SETC '8R' (2.1)	
& R	EN EQUÍNCA DE NOTA A TRANSPORTA EN ENTRE EN CONTRACTOR DE PORTO DE LA CONTRACTOR DE LA CONT	00001400
NEXT	ANOP	00001500
εı	SETA SI+1	00001600
	AIF (EI LE ELAST).LOOP	00001700
. SPACE	SPACE	00001800
• DE MCT		00001900
0.5.00.00	A MEXIT CALL	00002000
. BADREG	- OSISTISI (OT)	00002100
	AGO NEXT	C0002200
	The <b>MEND</b> of the control of the con	00002300
		0000200

## ABAL

	MACRO				00 0 00 1 0 0
ENAME	ABAL	εράμε στο Na nyeybe			00000200
	GBLB	<b>ELIB</b>			00000250
	AIF	(ELIB) .OK			00000300
		4. ABAL MACRO ILLE	AL FROM IN	TRINSIC*	00000400
	MEXIT				00000500
• OK	AIF	(D & EP) . SKIP			00000600
	EXTEN	$\mathcal{E} P_{i,j}$ , in the strength of the $i$		19 A 2 . 14 I • 1	00000700
.SKIP	ANOP	요즘 뭐 하나는 그들의 방향한다고 하			00000800
ENAME	BAL	4,8P CALI	. INTRINSIC		00000900
	MEND				00001000

# ACALL

	MACRO		00000100
ENAME	ACALL	. &P	00000200
	GBLB	ECALLOK	00000300
	AIF	(ECALLOK) . CALL	00000400
	MNOTE	12. ACALL OPTION NOT SPECIFIED IN AMAIN OR INTSIC=YES SPI	(00000500
		ECIFIED.	00000600
	MEXIT		00000700
. CALL	AIF	(D°EP).SKIP	00000800
	EXTRN	y SP in the second of the seco	00000900
SKIP	ANOP		00001000
ENAME	SCAL	0.8P CALL PROCEDURE ROUTINE	00001100
	MEND		00001200

# AERROR

erando de eng <b>macro</b> e estableció en el como del productivo de entre de entre de entre de entre el como de entre e	00000100
ENAME AERRCE ENUM, EGROUP=4	00000200
GBLA SERRONT, SERRNUMS (10), SERRGRPS (10)	00000300
ika dekanikiti. <b>LCIA</b> de <b>e</b> paratati da ka alaika tiraki ka alaan da ka istati da ka jarak da ka alaikita ka	00000400
AIF (ENUM GT 62).BADNUM	00000500
&I SETA SERRCNT	00000600
.DUFLOOP AIF (&I LE O) .NEWERR	00000700
'AIF (ENUM EQ EERRNUMS(EI) AND EGROUP EQ EERRGRPS(EI)). DUP	00000800
EI SETA EI-1	000003900
AGO .DUFLOOP	00001000
. NEWERR ANOP	00001100
SERRCNT SETA SERRCNT+1	00001200
EI SETA EERRCNT	00001300
SERRNUMS (SI) SETA ENUM	00001400
EERRGRPS (EI) SETA EGROUP	00001500
.DUP ANOP	G0001600
**************************************	** C0001700
ENAME SVC AERFOREI ISSUE SEND ERROR SVC	00001800
********* SEND ERROR SVC RETURNS CONTROL FOR STANDARD FIXUP*********	** 00001900
	00002000
BADNUM MNOTE 12, ERROR NUMBER GREATER THAN 62	00002100
	00002200

## ADATA

	MACKO	00100
		00 200
	GBLC &CSECT 000	C0300
******	*DATA CSECT***********************************	00400
	ERREARMS 000	00500
EDCSECT	SETC *#L*. * ECSECT* (1,6)	00600
EDCSECT	CSECT IN THE SECOND SEC	00700
	HEND THE PROPERTY OF THE PROPE	00800

## ACLOSE

		00000100
	MACKO	00000200
	ACLCSE	00003300
	GHLA SENTCHT	00000400
	GBLB EINPUT (20), EOUTPUT (20)	00000500
	GBLC ENAMES (20)	00000500
	ERRPARMS	
ΙЗ	SETA 1	00000700
	AIF (EINPUT(EI)).INOK	00000800
. LOOP	TO THE TOTAL TOTAL TOTAL CULMPS (ET) !	00000900
	The same of the sa	00001000
.INOK	TOTAL TOTAL CONTRACTOR DOD CHAMPS (ET)	00001100
		00001200
OUTOK	ANOF the state of the control of the	00001300
ΣI	SETA EI+1	00001400
	AIF (&I LE EENTCHT) . LOOP	00001500
	END CONTRACTOR OF THE CONTRACT	00001600
	MEND	00001000

## FRRPARMS

```
MACEC
                                                                  00000100
        ERRPARMS
                                                                  00000200
        GBLA SERRCNT, SERRNUMS (10), SERRGRPS (10)
                                                                  00000300
             EDCNE
        GBLB
                                                                  00000400
        GBLC
             SCSECT
                                                                  00000500
        LCLA
             εI
                                                                  00000600
        LCLC &S
                                                                  00000700
        AIF
              (EDONE) . MEND
                                                                  00000800
EDONE
        SETB 1
                                                                  00000900
        LTCRG
                                                                  00001000
00001100
        AIF
              (SERRCNT EQ 0) . NOERROR
                                                                  00001200
        SETC '#L'.' ECSECT' (1,6)
SECSECT
                                                                  00001300
SECSECT
        CSECT
                                                                  00001400
              (SERRCNT EQ 1) . MSG
        AIF
                                                                  00001500
εs
        SETC 'S'
                                                                  00001600
.MSG
        MNCTE **** SCSECT SENDS THE FOLLOWING ERRORSS.
                                                                  00001700
. LOCP
        ANOP
                                                                  00001800
ET
        SETA SI+1
                                                                  00001900
        SPACE 2
                                                                  00002000
        MNOTE **** ERROR NUMBER SERRNUMS (SI) IN GROUP SERRGRPS (SI) .
                                                                  00002100
        SPACE 1
                                                                  00002200
AERROREI DC
             H . 201
                           SVC CODE POR SEND ERROR
                                                                  00002300
        DC
             Y (SERRGRPS (SI) *256+SERRNUMS (SI)) 8 BIT GROUP AND NUMBER 00002400
        AIF
            (EI LT SERRCNT) . LOOP
                                                                  00002500
             . CCMMON
                                                                  00092600
.NCERROR MNOTE **** NO ERRORS SENT IN ECSECT*
                                                                  00002700
. COMMON ANOP
                                                                  00002800
00002900
. MEND
        MEND
                                                                  00003000
```

## WORKAREA

	MACRC						00 000 100 00 000 200
	WORKAREA GBLB ENCEXTRA						00000300
& NOEXTRA	SETB 1 NO ADDITIONAL	STACK ST	TORAGE KE	QUIRED FOR	THIS	ROUTINE	00000500
	MEND						

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

## 5.3 Library Routine Descriptions

This section contains descriptive material for all routines in the HAL/S-FC runtime library. The routines have been grouped into seven categories. The routines within each category are described in one subsection as follows:

- 5.3.1 Arithmetic
- 5.3.2 Algebraic
- 5.3.3 Vector/Matrix
- 5.3.4 Character
- 5.3.5 Array Functions
- 5.3.6 Miscellaneous
- 5.3.7 Remote Operations

The documentation is based upon the "load module" as a basic unit. A load module is the entity created by a single invocation of the AP-101 linkage editor. It has a primary member name and may have up to 16 alias names. The primary and alias names indicate entry points to the module.

For each load module in the runtime library, and LRD form will be found in the succeeding sections. The basic LRD form is shown in Figure 1. The circled numbers in the figure are explained below.

- O The boxed area of the form ( -7 below) contains information relating to qualities and attributes of the load module apart from any of its entry points.
- 1 In the upper right portion of every routine or entry point description, the name of the primary entry point will be seen. This serves as a quick reference aid in locating the documentation for a load module.

- 2 Source Member Name The name of the member in the assembler language source PDS of the library. This name is always the same as the primary entry point name.
- 3- Size of Code Area Each library module contains one code CSECT, regardless of the number of entry points. This number is the count of halfwords of code that would be used if the module were loaded. A module will be loaded if any one of its entry points is referenced.
- 4- Stack requirement If a module is not an intrinsic (see 6), it will have a requirement for runtime stack space. The minimum required will be one standard stack frame (18 Hw). The number listed on the form indicates the module's total stack requirement. If the module is an intrinsic, zero will be indicated. Individual entry points in one module cannot have different stack requirements. Therefore, the stack requirement is an attribute of the module.
- (5) Data CSECT size If the module contains a #L CSECT, its size is indicated. Otherwise, a zero is indicated. This number shows the number of halfwords of data area that will be used if the module is loaded.
- (6) Intrinsic/Library The appropriate box is marked. Entry points in a module are either all intrinsic or all library, hence this is a quality of the module. Sector 0 routines are noted here.
- 7- Other modules referenced A list of other load modules referenced in EXTRN statements by this load module. If this module is loaded, the indicated modules will also be loaded.
- (8) Entry point descriptions Following the aggregate attributes of the module in 0-7 above, the descriptions of specific entry points follow.
- (9) Primary Entry Name The name of the code CSECT\* in the module and the primary entry for the module in the library load module PDS.
- \* ENTRY label in the case of Sector 0 routines.

- 10 Function A brief prose description of what this entry point does.
- 11) Invoked By Entry points may be referenced directly from compiler-emitted code, from other library modules, or both. The appropriate boxes are marked. If the upper box is marked, an example of a HAL/S construct which results in reference to the entry point is shown. If the lower box is marked, the names of other modules which refer to this entry point are listed. If any of the other modules listed here are loaded, this module will also be brought in.
- 12 Execution Time The time, in microseconds, needed to perform this entry point's function. These times are obtained from examinations of trace listings of simulations of the execution of the particular library routine or entry point on Version 11.3 of the GPC simulator in detailed timing mode. Times include times for referenced routines unless specifically stated.
- 13 Input Arguments The data that the entry point receives as input is listed. "Type" indicates the nature of the data (integer, scalar, etc.).

  "Precision", where applicable, is generally SP for single precision and DP for double precision.

  "How Passed" indicates the method of communication of the data. In the case of DP scalar arguments, this field may indicate the first floating point register of an even/odd pair. "Units", when applicable, specifies the units presumed for an argument.
- 0utput Results The data that is considered the "answer" from the entry point. The fields are used in the same way as in (13).
- 15 Errors Detected If invocation of this entry point can result in a Send Error SVC being executed, the error #, cause, and standard fixup for all such errors are indicated.
- (16) Comments Any special behavior of this entry point or notes to users are entered here.
- (17) Algorithm The steps taken by the entry point to produce its results are shown. When appropriate, references are made to other entry point descriptions for further documentation.

In addition to the basic LRD form of Figure 1, which documents module attributes and the primary entry point, an extension LRD form is used to document additional alias entry points within a module. The extension LRD is shown in Figure 2. The circled numbers are explained below:

- 18 The primary entry name of the module is displayed. This is the same name as is displayed in the basic LRD form (1) to which this extension form is appended.
- 19 Secondary Entry Name The name of the secondary entry point being documented.
- 20 The remainder of the extension form is identical to the primary entry point description entries through (1), and describe the function and interface to this entry.

② ;					
<b>2</b> ) ,				①	
(2)	HAL	'S-FC LIBRARY	ROUTINE DESCRI	PTION	
•	Source Member Name:	And the constraints of the state of the stat	3 Size of Code	Area:	Hw
	Stack Requirement:				. Hw
	[] Intrinsic	6	☐ Procedur	: <b>e</b>	
<b>7</b>	Other Library Modules	Referenced:			
	ENTRY POINT DESCRIPTION				
<b>9</b> 1	Primary Entry Name:	er vanteres ( over 1981 1985 in the side of the side o			
10 F	Function:				
11) 1	Invoked by:  Compiler emitted	code for HAI	L/S construct of	the form;	
	Other Library Mod	lules:			
(12) I	Execution Time (micros	seconds):			
13 1	Input Arguments: Type	Precision	<u>.</u>	low Passed	Units
(14)	Output Results: <u>Type</u>	Precision		low Passed	<u>Units</u>
(15) I	Errors Detected: Error #		Cause	<u>Fi</u>	xup
(16)	Comments:				
$\sim$	Algorithme				

Figure 1: Basic LRD Form

_					
	`				
/1 c	• • •				
(1.8	5 )				
\	• ,		 _	_	_

Secondary Entry Name	:	en e		•	
Function:					
Invoked by:					
Compiler emitte	d code for HAL	/S construct	of the form:		
Other library m	odules:				
Execution Time (micr	oseconds):				
Input Arguments: Type	Precision		How Passed		Units
Output Results: Type	Precision		How Passed		<u>Units</u>
Errors Detected: Error #		<u>Cause</u>		<u>Fixup</u>	
Comments: Algorithm:					

Figure 2: Extension LRD Form

The following table shows the routines which are assigned to each group. The table contains a list of primary and secondary entry points with each secondary indented under its primary entry. With each primary entry point, basic descriptive information is shown along with the sizes of the csects in the module and the module's stack requirement. A final entry shows the timing information for the entry point. Secondary entry points have only the descriptive information and the timing for the entry. In cases where the timing information is too involved to be listed in the space available, the notice "See LRD" indicates that the detailed write-up of the module (on an LRD form in the proper subsection) should be referenced. In all cases, information in the table is taken from the LRDs and further details on the routines' performance can be found in those detailed descriptions.

## ARITHMETIC ROUTINES (Section 5.3.1)

ENTRY	FUNCTION	PREC.	CODE	DATA	STACK	TIME
DMOD	MOD (D,D)	D	42	2	0	74.6
DMDVAL	MIDVAL(D,D,D)	<b>D</b>	20	0	18	41.4
EMOD	MOD(S,S)	S	36	2	0	46.6
IMOD	MOD(I,I)	I	20	2	0	29.4
HMOD	MOD (H,H)	${f H}$				29.4
IREM	REMAINDER (I,I)	<b>I</b>	14	2	0	27.0
HREM	REMAINDER (H,H)	H				27.0
ROUND	ROUND(S)	Ι	84	2	0	39.0
CEIL	CEILING(S)	I				See LRD
DCEIL	CEILING(D)					See LRD
DFLOOR	FLOOR(D)	I				See LRD
DROUND	ROUND (D)					33.8
DTOI	$s \rightarrow r$	<b>I</b> .				33.8
DTRUNC	TRUNCATE (D)	$\mathbf{r}$				28.6
ETOI	$s \rightarrow I$	<b>I</b>		OR		39.0
FLOOR	FLOOR(S)	I		IGI PR(		See LRD
TRUNC	TRUNCATE (S)	I		NAC DCC		31.4
				CII		
				REPRODUCIBILITY ORIGINAL PAGE I		
				SI B		
				d S		
				OF THI POOR		
				RO(		
		医二氯磺基甲酚 医电影 医电影				

DATA

2

2

STACK

24

20

TIME

See LRD

See LRD

See LRD

345.2

302.2

See LRD

CODE

102

36

64

164

94

0

26

30

22

	ASTNH	ARCSINH(S)	S	64	0	20	See LRD
	ATANH	ARCTANH(S)	S	58	2	18	See LRD
	DACOS	ARCCOS (D)	D	150	2	18	See LRD
	DASIN	ARCSIN(D)	D				See LRD
	DACOSH	ARCCOSH (D)	D	42	2	22	See LRD
	DASINH	ARCSINH (D)	D	94	0	22	See LRD
	DATANH	ARCTANH (D)	D	90	2	18	See LRD
ហ	DATAN2	ARCTAN2 (D,D)	D	194	26	18	248.4
39	DATAN	ARCTAN (D)	D				237.3
	DEXP	EXP (D)	D	154	66	18	
	DLOG	LOG(D)	<b>D</b>	140	2	22	290.5
	DPWRD	D**D	D	38	4	22	282.2
	DPWRI	$D_{\star\star}I$	D	40	2		See LRD
	DPWRH	D**H	D		2	18	See LRD
	DSIN	SIN(D)	D	102	62		See LRD
	DCOS	COS (D)	D	102	62	20	267.0
	DISNH	SINH(D)		* ^ ^			261.8 - 264.2
			D.	130	2	22	See LRD
	DCOSH	COSH(D)	D				422.6
	DSQRT	SQRT (D)	D	70	2	26	345 2

PREC.

S

S

S

S

D

D

ENTRY

ASIN

ACOS

ACOSH

ASINH

DTAN

DTANH

FUNCTION

ARCCOS (S)

ARCSIN(S)

ARCCOSH(S)

ARCSINH(S)

TAN(D)

TANH (D)

ENTRY	FUNCTION	PREC.	CODE	DATA	STACK	TIME
EATAN2	ARCTAN2(S,S)	s	132	10	18	120.0
ATAN	ARCTAN(S)	S				116.5
EPWRE	S**S	s	^32	4	22	See LRD
EPWRI	S**I	s	38	2	18	See LRD
EPWRH	S**H	s				See LRD
EXP	EXP(S)	S	108	2	18	141.8
IPWRI	I**I	I	46	2	18	See LRD
HPWRH	H**H	H				See LRD
IPWRH	I**H	I				See LRD
LOG	LOG(S)	S	90	2	18	140.5
SIN	SIN(S)	S	70	30	0	123.6 - 124.5
cos	cos(s)	S				122.1 - 123.1
SINH	SINH(S)	S	80	2	18	See LRD
COSH	COSH(S)	S				228.9
SQRT	SQRT(S)	S	48	14	0	88.3
TAN	TAN(S)	S	112	4	20	164.0
TANH	TANH (S)	S	56	0	18	See LRD

ENTRY	FUNCTION	SIZE	PREC.	CODE	DATA	STACK	TIME
MM0DNP	Scalar to Partitioned Matrix Move	n,m	D	12	0	0	6.8+n(4.0+8.0m)
MM0SNP		n,m	S	10	0	0	6.4+n(4.4+6.4m)
MMLDNP	Partitioned Matrix Move	n,m	D	18	0	0	10.8+n(5.4+12.2m)
MMLSNP	경기의 결혼하고 그렇게 되었다. 그는 내가 그 말았다. 중심한 말로 가는 그런 그렇게 되었다. 그는 그 것이 있다.	n,m	s	16	0	0	10.8+n(5.4+9.4m)
MMLTNP		n,m	D-S	16	0	0	10.4+n(5.8+10.6)
MMLWNP		n,m	S-D	18	0	0	13.6+n(5.0+11.0m)
MM6DN	Matrix Multiply	(m,n),(n,l)	D D	42	0	0	22.2+m(10.8+l(21.2+27.n))
MM6D3		(3,3),(3,3)	D	32	0	0	671.6
MM6SN		$(m,n)$ , $(n,\ell)$	S	40.	0	0	22.2+m(10.8+l(20.2+18.0n))
MM6S3		(3,3),(3,3)	s	24	0	0	409.6
MMllDN	Matrix Transpose	n,m	D	16	0	0	8.0+m(5.8+12.2n)
MM11D3	공격회의 경류는 이렇게 하는데 하는데	3,3	D D	22	0	C	93.6
MMllsn		m,n	s	16	0	0	8.4+m(5.8+9.4n)
MM11S3		3,3	s	18	0	0	71.8
MM12DN	Matrix Determinant	n,n	D	150	0	22	See LRD
MMl2D3		3,3	D	44	0	18	229.6
MMl2SN		n,n	s	138	0	20	See LRD
MMl2S3		3,3	S	26	0	18	116.0
MM13DN	Matrix Trace	n,n	D	10	0	0	12.0+10.2n
MM13D3		3,3	D	8	0	0	19.8
MM13SN		n,n	s	8	0	0	8.8+6.2n
MM13S3		3,3	S	4	0	0	9.8

5-4

	ENTRY	FUNCTION	SIZE	PREC.	CODE	DATA	STACK	TIME
	MM14DN	Matrix Inverse	n,n	D	258	2	20	63.0+129.5n+43.0n <sup>2</sup> +65.4n <sup>3</sup>
	MML4D3		3,3	D	128	2	18	795.4
	MM14SN		n,n	S	242	2	20	52.0+39.2n+10.5n <sup>2</sup> +54.6n <sup>3</sup>
	MM14S3		3,3	s	80	2	18	458.8
	MM15DN	Identity Matrix	n,n	D	18	0	0	15.6+5.0n+11.2n <sup>2</sup>
	MM1.5SN		n,n	S	14	0	0	10.0+5.2n+9.6n <sup>2</sup>
	MM17D3	Matrix to a Power	3,3	D	86	0	20	See LRD
	MM17DN	[발발되었다] 때 경우 상태 일시점	n,n	D D				See LRD
	MM17S3		3,3	s	78	0	20	See LRD
យា	MM17SN		n,n	s				See LRD
5-42	MV6DN	Matrix times Vector	(m,n),n	D	24	0	0	12.0+m(19.3+26.0n)
	MV6D3		(3,3),3	D	22	0	0	304.4
	MV6SN		(m,n),n	s	18	0	0	11.2+m(11.0+18.4n)
	MV6S3		(3,3),3	S	20	. 0	0	137.6
	VM6DN	Vector times Matrix	n,(n,m)	D	26	0	0	23.2+m(23.2+27.6n)
	VM6D3	살이 많이 들어 그 그는 사람이 있다.	3,(3,3)	D	24	0	0	227.8
	VM6SN		n,(n,m)	S	22	0 0	0	12.4+m(19.2+18.2n) Q
	VM6S3		3, (3,3)	s	16	0.0	0	141.2
	VO6DN	Vector Outer Product	n,m	D	20	0	0	12.8+n(5.8+24.4m)
	VO6D3	기회의 설립 스크리 및 기회의 스크리 스키 하는 기회 (대학교) 이 등에는 설립 (기학의 기본 학교)	3,3	D	22	0	0	251.0
	VO6SN	보이고함 어떻게 맞다고 된 이렇다	n,m	S	20	0	<b>O</b> , .	12.4+m(19.2+18.2n) 141.2 12.8+n(5.8+24.4m) 251.0 14.2+n(5.8+14.4m) 160.6
	VO6S3		3,3	S	20	0	0	
	VV0DN	Scalar to Vector Move	n	D	6	0	0	7.0+5.1n
	VVODNP S	Scalar to Column Vector Move	n	D	6	0	0	7.0+7.2n
							· · · · ·	<b>○</b> +

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

ENTRY	<u>FUNCTION</u>	SIZE	PREC.	CODE	DATA	STACK	TIME
vvosn	Scalar to Vector Move	n	S	6	0	0	7.0+5.6n
VV0SNP	Scalar to Column Vector Move	n	S	6	0	0	7.0+6.0n
VVIDN	Vector Move	n	. D	. 8	0	0	4.2+10.2n
VV1D3			D	14	0	0	25.2
VV1D3 VV1D3P	Column Vector Move	3	D	18	0	0	See LRD
VVIDN		n	D				See LRD
VVIDN	. Vector Move	n	S	8	0	0	4.2+7.8n
VVISN VVIS3	vector move	3	S	8	0	0	16.8
	Column Vector Move	3	S	14	0	0	See LRD
VV1S3P			S	•••	, , <b>,</b>		See LRD
VVISN		n		8	0	0	4.2+9.0n
VVlTN	Vector Move	n	D-S			_	21.2
VV1T3		3	D-S	12	0	0	See LRD
VV1T3P	Column Vector Move	3	D-S	14	0	. 0	See LRD
VVlTN	B: [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [	n	.D-S				
VVlWN	Vector Move	n	S-D	10	0	0	8.4+9.0n
VV1W3	경염 희망성 열차 회사 기사는 사람들은	3	S-D	12	0	0	23.8
VV1W3P	Column Vector Move	3	S-D	18	0	0	See LRD
VVlWN		n	S-D				See LRD
VV2DN	Vector Add/Matrix Add	n	D	14	0	0	8.8+20.6n
VV2D3	Vector Add	3	D	22	0	0 0	51.4
VV2SN	Vector Add/Matrix Add	n	S	10	0	0	8.2+13.6n
VV2SN VV2S3	Vector Add	3	S	12	0 .	0	29.6
	Vector Subtract/Matrix Subtract	n	D	16	0	0	6.0+22.7n
MACAA	ASCROT DUDGERORY LIGHTEN DUDGEROR			+0	•	•	

5-43

# VECTOR/MATRIX ROUTINES (CONTINUED) (Section 5.3.3)

ENTRY	FUNCTION	SIZE	PREC.	CODE	DATA	STACK	TIME
<del></del> :	Vector Subtract	3	D	24	0	0	55.4
VV3D3		n	s	10	0	0	8.4+13.6n
VV3SN	Vector Subtract/Matrix Subtract	3	S	12	0	0	29.6
vv3s3	Vector Subtract	n	D	8	0	0	7.0+23.4n
VV4DN	Vector or Matrix Times Scalar		D	18	0	0	68.4
VV4D3	Vector Times Scalar	3		8	0	0	7.0+14.0n
VV4SN	Vector or Matrix Times Scalar	n	<b>S</b> -			0	38.4
vv4s3	Vector Times Scalar	3	S	12	0		37.0+24.2n
VV5DN 7	Vector or Matrix Divided by Scalar	n	D	16	2	0	98.4
VV5D3	Vector Divided by Scalar	3	D	26	2	0	7.2+18.0n
VV5SN V	Vector or Matrix Divided by Scalar	n	S	14	2	0	50.6
VV5S3	Vector Divided by Scalar	3	s	18	2	0	
VV6DN	Vector Dot Product	$\mathbf{n}$	D	12	0	0	16.4+25.4n
VV6D3	ĸŢĔĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸĸ	3	D	16	0	0	71.8
VV6SN		n	s	12	0	0	15.2+16.8n
VV6SA		3	s	10	0	0	41.8
VV7DN	Vector or Matrix Negate	n	D	8	0	0	7.0+11.4n
VV7DN VV7D3	Vector Negate	3	D	18	0 -	0	32.4
	Vector or Matrix Negate	n	s	8	0	0	7.0+9.0n
VV7SN	Vector Negate	3	S	12	0	0	23.4
vv7s3		3	D	12	0	0	See LRD
AA8D3	Vector Compare		D				See LRD
VV81		n	S	12	0	0	See LRD
VV8S3	Vector Compare	3		1.2		<b>.</b>	See LRD
VV8	SN Vector or Matrix Compare	n	S				

### VECTOR/MATRIX ROUTINES (CONTINUED) (Section 5.3.3)

ENTRY	<u>FUNCTION</u>	SIZE	PREC.	CODE	DATA	STACK	TIME
VVl0D3	Unit Vector	3 .	D	56	2	20	402.7
		n	D				259.7+47.8n
VV10DN	Mani tudo	n n	D				226.6+24.4n
VV9DN	Vector Magnitude	3	D				300.2
VV9D3		3	S	50	2	24	236.4
VV10S3	Unit Vector		S				130.6+32.8n
VV10SN		n n	-				118.9+14.0n
VV9SN	Vector Magnitude	n	S				
VV9S3	전환경 그런 이번 하는 사람이 되었다. (1982년 1982년 19 - 1982년	3	S				168.3
VX6D3	Vector Cross Product	3	D	36	0	0	137.6
<sup>5</sup> ∇X6S3		3	s	22	0	0	78.0

ENTRY	<u>FUNCTION</u>	CODE	DATA	STACK	TIME
CASPV	Partitioned Assign to VAC	64	<b>.</b>	0	See LRD
CASP	Partitioned Assign				See LRD
CASV	Assign to VAC	28	0	0	29.2 (n=0) See LRD
CAS	Assign				32.0 (n=0) See LRD
CATV	Catenate into VAC	76	0	0	See LRD
CAT	Catenate into Data				See LRD
CINDEX	INDEX Function	52	0	18	See LRD
CLJSTV	LJUST	40	2	18	See LRD
CPAS	Assign to Partition	80	2	20	See LRD
CPASP	Partition Assign to Partition	16	0	146	See LRD
CPR	Compare (= or ¬=)	46	0	0	See LRD
CPRC	Compare (all relations except = and ¬=)				See LRD
CPRA	Arrayed Compare	20	0	22	See LRD
CRJSTV	RJUST	46	2	18	See LRD
CTRIMV	TRIM	94	0	18	See LRD



### ARRAY ROUTINES (Section 5.3.5)

ENTRY	FUNCTION	PREC.	CODE	DATA	STACK	TIME
DMAX	MAX (DA)	D	10	0	0	See LRD
DMIN	MIN (DA)	$\mathbf{D}$	10	0	0	See LRD
DPROD	PROD (DA)	D	14	0	0	See LRD
DSUM	SUM (DA)	D	6	0	0	7.2+11.6n
EMAX	MAX(SA)	S	8	0	0	See LRD
EMIN	MIN (SA)	s	8	0	0	See LRD
EPROD	PROD (SA)	S	10	0	0	See LRD
ESUM	SUM(SA)	${f s}$	6	0	0	5.2+6.6n
HMAX	MAX (HA)	H	8	0	0	See LRD
HMIN	MIN (HA)	H	8	0	0	See LRD
HPROD	PROD (HA)	H	12	0	0	See LRD
HSUM	SUM(HA)	H	6	0	0	4.4+5.4n
IMAX	MAX(IA)	Ι	8	0	.0	See LRD
IMIN	MIN (IA)	I	8	0	0	See LRD
IPROD	PROD(IA)	I	22	0	0	See LRD
ISUM	SUM(IA)	I	6	0	0	4.4+5.4n

# MISCELLANEOUS ROUTINES (Section 5.3.6)

ENTRY	<u>FUNCTION</u>	CODE	DATA	STACK	TIME
втос	Bit to Character Conversion	28	0	0	161.0 (16 bits)
CSHAPQ	Shaping Function	40	4	18	See LRD
CSLD	SUBBIT Load of Character	246	4	22	See LRD
CSLDP	SUBBIT Load of Partitioned Character				See LRD
CPSLD	Partitioned SUBBIT Load of Character				71.8
CPSLDP	Partitioned SUBBIT Load of Partitioned Character				See LRD
CSSTP	SUBBIT Store to Partitioned Character				See LRD
CPSST	Partitioned SUBBIT Store to Character				114.4
CSST	SUBBIT Store to Character				See LRD
CPSSTP	Partitioned SUBBIT Store to Partitioned Character				See LRD
CSTRUC	Structure Compare	12	0	0	5.4+10.4n
СТОВ	Character to Bit Conversion	32	2	18	See LRD OR
CTOE	Character to SP Scalar Conversion	287	2	30	See LRD CRO
CTOD	Character to DP Scalar Conversion				See LRD
CTOI	Character to DP Integer Conversion	104	2	20	See LRD
СТОН	Character to SP Integer Conversion				See LRD
CTOK	Character to Bit Conversion, DEC Radix				See LRD OR See LRD SEE
					POOR THE

### MISCELLANEOUS ROUTINES (CONTINUED) (section 5.3.6)

ENTRY	FUNCTION	CODE	DATA	STACK	TIME
CTOX	Character to Bit Conversion, HEX Radix	58	4	18	See LRD
CTOO	Character to Bit Conversion, OCT Radix				See LRD
DSLD	SUBBIT Load of DP Scalar	22	2	18	36.5
DSST	SUBBIT Store into DP Scalar	54	2	18	64.6
ETOC	SP Scalar to Character Conversion	278	64	20	336.9
DTOC	DP Scalar to Character Conversion				602.5
ETOH	SP Scalar to SP Integer Conversion	14	0.	0	15.4
DTOH	DP Scalar to SP Integer Conversion				17.4
GTBYTE	Character Fetch	14	0	0	See LRD
ITOC	DP Integer to Character Conversion	104	0	28	254.6
HTOC	SP Integer to Character Conversion				189.6
ITOD	DP Integer to DP Scalar Conversion	20	0	0	15.6
ITOE	DP Integer to SP Scalar Conversion	6	0	0	12.0
KTOC	Bit to Character Conversion, DEC Radix	70	0	0	262.5 (16 bits)
MSTRUC	Structure Move	8	0	0	4.2+9.4n
OSHAPQ	Shaping Functions	74	0	18	42.6+31.8n
RANDOM	Random Number Generator, Uniform Di	st.46	2	18	54.4
RANDG	Random Number Generator, Gaussian D				575.8
STBYTE	Character Store	22	0	0	See LRD
XTOC	Bit to Character Conversion, HEX	68	0	0	See LRD
OTOC	Bit to Character Conversion, OCT Radix				See LRD

### REMOTE ROUTINES (Section 5.3.7)

	ENTRY	FUNCTION	CODE	DATA	STACK	TIME
Α.	CHARACTER ROUTINE					
	CASRPV	Partitioned Assign to VAC	86	2	22	See LRD
	CASRP	Partition Assign				See LRD
	CASRV	Assign to VAC	36	0	18	See LRD
	CASR	Assign				See LRD
	CPASR	Assign to Partition	132	2	24	See LRD
	CPASRP	Partition Assign to Partition	16	0	146	See LRD
		[발표] 전 바람이 되는 사람들이 없다.				
В.	STRUCTURE ROUTINE					
	CSTR	Structure Compare	18	0	18	See LRD
	MSTR	Structure Move	10	0	18	See LRD

50

c.	ENTRY VECTOR AND MAI	FUNCTION TRIX ROUTINES	SIZE	PREC.	CODE	DATA	STACK	TIME
	MR0DNP	Scalar to Partitioned Matrix Move	n,m	D	16	0	20	22.8+n(5.6+9.8m)
	MR0SNP		n,m	S	16	0	20	22.8+n(5.6+8.6m)
	MRIDNP	Partitioned Matrix Move		D	22	0	20	28.4+n(8.2+15.0m)
	MRlsnp	가 보통 그리는 이를 보고 하 <b>면</b> 하는 것 같은 사람들이 있다. 일본하고 있는 음식을 가는 것 같습니다. 그런 기계를 다 되었다.	n,m	s	22		22	28.4+n(8.2+12.6m)
	MRITNP		n,m	D-S	24	0	22	31.2+n(7.6+13.8m)
	MR1WNP VR0DN		n,m	S-D	24	0	22	32.8+n(8.2+15.8m)
ហ្	VRODNP	Scalar to Vector Move	n	D	6	0	18	16.4+9.2n
	Table to the second of the sec	Scalar to Column Vector Move	n	D	10	0	18	21.2+10.0n
	VR0SN	Scalar to Vector Move	n	S	6	0	7.0	
	VR0SNP	Scalar to Column Vector Move	n	S	10	0	18 18	16.4+8.0n 21.2+8.8n
	VR1DN	Vector Move	n	D	8	0	18	16.4+15.0n
	VRlDNP	Column Vector Move	n	D	20	0	18	See LRD
	VRLSN	Vector Move	n	s	8	0	18	16.4+12.6n
	VRLSNP	Column Vector Move	n	s	20	0	18	See LRD
	VRLTN	Vector Move	n	D-S	8 8	0	18	16.4+13.8n
	VRLTNP	Column Vector Move	n	D-S	20	0	18	See LRD
	VR1WN	Vector Move	n	S-D	10	0	18	20.6+13.8n
	VRLWNP	Column Vector Move	n	S-D	22	0	18	See LRD

Ü

#### 5.3.1 Arithmetic Routine Descriptions

This subsection presents the detailed descriptions of a class of routines generally denoted as "Arithmetic". Appendix C of the HAL/S Language Specification contains a list of HAL/S functions which are implemented by the routines described here.

				Е	MOD	
	HAL/	S-FC LIBRARY	ROUTINE DESCR	RIPTION		
. 5	Source Member Name:	DMOD	Size of Code	Area:	42	HW
5	Stack Requirement:	0 Hw	Data CSECT S	ize:	2	Hw
	▼ Intrinsic		Proced	ure		
•	Other Library Modules	Referenced:	None			
Ī	ENTRY POINT DESCRIPTIO	NS				
J	Primary Entry Name:	DMOD				
	Function: Calculates	HAL/S MOD f	unction in dou	ble precis	sion.	
	Invoked by:  Compiler emitted  MOD(A,B), wher					calar.
	Other Library Mod	lules:				
	Execution Time (micros	econds):	74.6	REPROD ORIGINA	ÜCIBILITY OI AL PAGE IS I	THE
	Input Arguments:				.5	77 d
	<u>Type</u>	Precision		How Passe	<u>:a</u>	Uni
	Scalar Scalar	DP DP		F0/F1 F2/F3		
	Output Results:					
	Type	Precision		How Passe	<u>:d</u>	Uni
	Scalar	DP		FO/F1		<del></del> .
	Errors Detected:					
	Error #		Cause		Fixup	
	19	mod doma	in error when A < 0	B=0,	Return A	
	Comments: If one argu	ment is seve	eral orders of	magnitude	greater than	the

Comments: If one argument is several orders of magnitude greater than the other argument, the code sequence for A-(|B|\*FLOOR(A/|B|) loses some bits of precision... it is possible to have the result of A(mod B) be greater than |B|.

For arguments that are orders of magnitude apart, it is necessary to check that the result is  $\leq$  |B|. If RESULT > |B|, return |B|.

DMOD

Comments (Cont'd.)

Registers Unsafe Across Call: R4,F0,F1,F2,F3,F4,F5.

and the second				VGMC	AL	
	HAL,	/s-FC LIBRAR	Y ROUTINE DES	CRIPTION		
Source Mer	mber Name:	DMDVAL	Size of Co	de Area:	20	_ Hw
Stack Requ	irement:	18 Hw	Data CSECT	Size:	00	Hw
	☐ Intrinsic	3	R Proce	edure		
Other Lib	rary Modules	Referenced:	None			
ENTRY POI	NT DESCRIPTION	ONS				
Primary E	ntry Name: _	DMDVAL	·····			
Function:		mid value uments.	of three do	ouble precis	ion scal	ar
Invoked by Comp		code for HA	L/S construct	of the form:		
-				uble precis		ars.
		a				
Otne	r Library Mo	dules:				
Execution	Time (micro	seconds):	41.4			
Input Arg		Precision		How Passed		Units
scala	<del></del>	DP		F0		
scala	1.77	DP		F2		
scala <b>Output</b> Re	r enlte:	DP		F4		
Typ		Precision		How Passed		Units
scala	<del></del>	DP		F0		
Errors De	tected:					
<u>Err</u>	or #		Cause		Fixup	
Comments:						
	sters Unsafe	Across Call	: F0,F1,F2,F	3,F4,F5.		
				3,F4,F5.		
Regi	if A =	B THEN RET	rurn A;	3,F4,F5.		
Regi	IF A =	B THEN RET	rurn A;	3,F4,F5.		
Regi	IF A = IF A < IF B ELSE	B THEN RET B THEN DO S <= C THEN LIF A <= C	TURN A; N RETURN B; C THEN RETUR			
Regi	IF A = IF A < IF ELSE	B THEN RET B THEN DO	TURN A; N RETURN B; C THEN RETUR	RN C;		
Regi	IF A = IF A < IF B ELSE	B THEN RET B THEN DO S <= C THEN L IF A <= C SE RETURN	TURN A; N RETURN B; C THEN RETUR	RN C;	GWAL PA	English and the state of

ELSE IF C <= A THEN RETURN C;

ELSE RETURN A;

END;

•	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
Source Member N	ame: EMOD	Size of Code Area:	36 Hw
Stack Requireme	nt: 0 Hw	Data CSECT Size:	2 Hw
X In	trinsic	☐ Procedure	
Other Library M	odules Referenced: _	None	-
ENTRY POINT DES	CRIPTIONS		
Primary Entry N	ame: EMOD		
Function: Calcu	lates HAL/S MOD fun	ction in single precision.	
MOD (A,B)	, where A and B are	S construct of the form: single precision scalars.	
MOD (A,B)	, where A and B are ary Modules:	single precision scalars.	
MOD(A,B)  Other Libra  Execution Time  Input Arguments	, where A and B are ary Modules: (microseconds): 46	single precision scalars.	
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type	, where A and B are ary Modules: (microseconds): 46. : Precision	single precision scalars.	<u>Un</u>
MOD(A,B)  Other Libra  Execution Time  Input Arguments	, where A and B are ary Modules: (microseconds): 46	single precision scalars.	<u>Un</u>
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type  Scalar  Scalar	, where A and B are ary Modules:  (microseconds): 46.  Precision SP	single precision scalars.  6  How Passed  F0	Un
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type  Scalar  Scalar	, where A and B are ary Modules:  (microseconds): 46.  Precision SP	single precision scalars.  6  How Passed  F0	<u>Un</u>
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type  Scalar  Scalar  Scalar  Output Results:	where A and B are ary Modules:  (microseconds): 46.  Precision SP SP	single precision scalars.  6  How Passed F0 F2	
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type Scalar Scalar  Output Results: Type Scalar	where A and B are ary Modules:  (microseconds): 46.  Precision SP SP Precision SP	FO F2	
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type Scalar Scalar Scalar Output Results: Type	where A and B are ary Modules:  (microseconds): 46.  Precision SP SP Precision SP	FO F2	
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type Scalar Scalar  Output Results:  Type Scalar  Scalar  Errors Detected:	where A and B are ary Modules:  (microseconds): 46.  Precision SP SP Precision SP	How Passed F0 F2 How Passed F0 F0 F2 Cause	<u>Un</u>
MOD(A,B)  Other Libra  Execution Time  Input Arguments  Type Scalar Scalar Output Results: Type Scalar Errors Detected: Error #	where A and B are ary Modules:  (microseconds): 46.  Precision SP SP Precision SP	How Passed F0 F2 How Passed F0 F0 F2 Cause	<u>Un</u>

<del>,</del>			IMOD	
	HAL/S-FC LIBRA	RY ROUTINE DESCRIPTION	rangan dan kacamatan dan salah s Managan dan salah sa	
		Size of Code Area:		Hw
	Source Member Name:IMOD	-	20	- 17.44
	Stack Requirement: 0 Hv	Data CSECT Size: _	2.	– <sup>Hw</sup>
	X Intrinsic	<pre>Procedure</pre>		
	Other Library Modules Referenced	None.		
	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: IMOD			
	Function: Calculates A Mod(B). If MOD= B - A-(n+1) B   where n is at A < 0, MOD= B - A+(n-1) B  , where Invoked by:  X Compiler emitted code for His MOD(A,B), A and B are both	n integer and [A-(n+1) e n is an integer and AL/S construct of the	B  < 0 < $ A-n $ $ A-n $ $ A-n $ $ A-n $ form:	B ]. Fo   <[A+n B
	fullword integer value.			
	Other Library Modules:			
	Execution Time (microseconds):	29.4		
	Input Arguments: Type Precision	How Pa	esed	Units
	Type Precision Integer DP	R5		-
	Integer DP	R6		·
	Output Results:	How Pa	bossa	Units
	Type Precision Integer DP	R5		UIIICS
	Errors Detected:			
	Error #	Cause	Fixup	
		ot defined for first	Return firs	st arg.
	arg <	0 and second arg = 0.		
	Comments: Registers Unsafe Across	Call: R2,R4,R5,R6,R7		
	Algorithm:  Use $ B $ . If B=0, then check for If A < 0, error. For B $\neq$ 0, the depending on the value of A. For A < 0, MOD(A,B) = A- $ B $ *(A/ $ B $ )+  always non-negative.  For A > 0, MOD = REMAINDER(A,B)	mod is found using one $A \ge 0$ , MOD(A,B) = A-[B]. For all values of	of two formulae $ B  \star (A/ B )$ . Find the result is	€,
	AP-101 division (scalar or intege			

IMOD

Secondary Entry Name	: HMOD				
Function: Performs E precision	MAL/S MOD(A,B) wintegers.	here both I	A and B are sing	le	
Invoked by:					
X Compiler emitte	d code for HAL/	'S construc	t of the form:		
	e A and B are b			ers.	
Other library m	odules:				
Execution Time (micr	coseconds): 29.	4			
Input Arguments:					
Type	Precision		How Passed		Units
Integer	SP		R5 R6		_
Integer	SP				
Output Results:					
Type	Precision		How Passed		Units
Integer	SP		R5		<del>-</del>
Errors Detected: Error #		Cause		Fixup	
Same as IMOD					
Comments:	nsafe Across Cal	ll: R2,R4,	R5,R6,R7.		
Algorithm:					
Same as IMOD					

REPRODUCIBILITY OF THE ORIGINAL PAGE IN POOR

			IREM	-
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name	: IREM	Size of Code Area:	14	Hv.
Stack Requirement:	0 Hw	Data CSECT Size:	2	Hw
X Intri	nsic	☐ Procedure		
Other Library Modu	les Referenced: _			
ENTRY POINT DESCRI	PTIONS			
Primary Entry Name	: IREM			
Function: Calcul	ates integer remai	inder of (A/B).		
of A or B  Other Library	is double precision	are both integers and	i at least on	ie
of A or B	is double precision Modules:	on: "	i at least on	<b>1e</b>
of A or B Other Library	is double precision Modules:	on: "	i at least on	<b>1e</b>
of A or B Other Library  Execution Time (mid Input Arguments: Type	is double precision  Modules:  croseconds): 27	.0 How Pass		
of A or B Other Library  Execution Time (mid	is double precision Modules: croseconds): 27	.0 How Pass		
of A or B  Other Library  Execution Time (mid  Input Arguments:  Type  Integer	is double precision  Modules:  croseconds): 27	.0  How Pass a be SP		ne ()
of A or B Other Library  Execution Time (mid Input Arguments:  Type Integer Integer Output Results:  Type	is double precision  Modules:  croseconds): 27	.0  How Pass a be SP	<u>ed</u>	
of A or B  Other Library  Execution Time (mid  Input Arguments:  Type Integer Integer Output Results:	is double precision  Modules:  croseconds): 27  Precision  DP  DP  OP  DP  OP  DP  OP  DP  OP  DP  OP  DP  OP  DP  D	.0  How Pass R5 R6	<u>ed</u>	<u> </u>
of A or B Other Library  Execution Time (mid Input Arguments:  Type Integer Integer Output Results:  Type	is double precision  Modules:  croseconds): 27  Precision  DP one can  Precision	.0  How Pass R5 R6  How Pass	<u>ed</u>	
of A or B Other Library  Execution Time (mid Input Arguments:  Type Integer Integer Cutput Results:  Type Integer Errors Detected: Error #	is double precision  Modules:  croseconds): 27  Precision  DP one can  Precision  DP DP	.0  How Pass R5 R6  How Pass	<u>ed</u>	<u> </u>
of A or B Other Library  Execution Time (mid Input Arguments:  Type Integer Integer Cutput Results:  Type Integer Errors Detected:	is double precision  Modules:  croseconds): 27  Precision  DP DP one can  Precision  DP DP	.0  How Pass R5 R6  How Pass R5	<u>ed</u>	<u> </u>
of A or B Other Library  Execution Time (mid Input Arguments:  Type Integer Integer Output Results:  Type Integer Errors Detected: Error # 16	is double precision  Modules:  croseconds): 27  Precision  DP DP one can  Precision  DP DP	.0  How Pass R5 R6  How Pass R5 R6	ed Fixup	<u> </u>
of A or B  Other Library  Execution Time (mid  Input Arguments:  Type Integer Integer Coutput Results:  Type Integer Errors Detected: Error #  16  Comments:	is double precision  Modules:  croseconds): 27  Precision  DP DP one can  Precision  DP DP	.0  How Pass R5 R6  How Pass R5 Cause Lenominator (B)	ed Fixup	<u> </u>

If B=0, then error. For  $B \neq 0$ , the remainder is found using REMAINDER(A,B) = [A-B\*(A/B)]. The result can be negative.

IREM

Secondary Entry Name:	HREM		
Function: Calculates	integer remainder o	of A/B.	
Invoked by:			
X Compiler emitted REMAINDER(A,B)	d code for HAL/S con, where A and B are	nstruct of the form: both single precision inte	gers.
Other library mo	odules:		
Execution Time (micro	oseconds): 27.0		
Input Arguments:			
Type	Precision	How Passed	Units
Integer	SP	R5	
Integer	SP	R6 86 1	
Output Results:			
Type	Precision	How Passed	Units
Integer	SP	R5	
Errors Detected:			
Error #	Cau	se Fix	ap
16	zero denom	ination (B) Retur	<del></del>
Comments: Registers	Unsafe Across Call:	R2,R4,R5,R6,R7.	
Algorithm:			
Same as IREM.			

			R	OUND	
		AL/S-FC LIBRARY ROUTIN	E DESCRIPTION		
	Source Member Name:	ROUND Size	of Code Area:	84	Hw
	Stack Requirement:	0 Hw Data	CSECT Size:	. 2	Hw
	X Intrin	nsic SECTOR 0	Procedure		
	Other Library Modul	es Referenced: No	ne		
-	ENTRY POINT DESCRIP	PTIONS			
	Primary Entry Name:	ROUND			
	Function: Convert	s single precision scal	ar to fullword i	integer.	
	Invoked by:	ted code for HAL/S cons	truct of the fo	rm:	
		e X is a single precisi		7-1	
	1100112 (11)				
	X Other Library	Modules: QSHAPQ			
	X Other Library	Modules: QSHAPQ			
	X Other Library Execution Time (mid				
	Execution Time (mid	croseconds): 39.0	How Page		Units
	Execution Time (mid Input Arguments: Type		How Pass	<u>ed</u>	Units
	Execution Time (mic	croseconds): 39.0	How Pass	ed	Units
	Execution Time (mident Input Arguments:  Type  Scalar	croseconds): 39.0  Precision		<b>ed</b>	Units
	Execution Time (mid Input Arguments: Type	croseconds): 39.0  Precision			Units _ Units
	Execution Time (mid Input Arguments:	eroseconds): 39.0  Precision  SP	FO		
	Execution Time (mident Input Arguments:  Type  Scalar  Output Results:  Type	Precision  Precision  Precision	FO How Pass		
	Execution Time (mid Input Arguments:	Precision  Precision  Precision	FO  How Pass  R5	ed Fixup	Units
	Execution Time (mid Input Arguments:	Precision SP  Precision DP  Cause Scalar to lar	How Pass R5 ge for Set	ed Fixup to max/min r	Units
	Execution Time (mid Input Arguments:	Precision SP Precision DP Cause	How Pass R5 ge for Set ersion. N	Fixup to max/min r ue: egmax = X'800	Units - cepresenta
	Execution Time (mid Input Arguments:	Precision SP  Precision DP  Cause Scalar to lar	How Pass R5 ge for Set ersion. N	ed  Fixup  to max/min rue:	Units - cepresenta
	<pre>Execution Time (mid Input Arguments:</pre>	Precision SP  Precision DP  Cause Scalar to lar	How Pass R5 ge for Set ersion. N P	Fixup to max/min r ue: egmax = X'800 osmax = X'7FF	Units - cepresenta

then routine merges into the double precision float-to-fix routine, DROUND.

ÞΩ	UND	
RO	UND	

Secondary Entry Name:	CEIL				
Function: Performs H  > the argu  Invoked by:	AL/S CEILING f ment.	function: Ret	urns small	est integer	
CEILING(X), w	there X is a s	/S construct o	of the form		
Execution Time (micro	oseconds): 33	1.4 if $x \ge 0$ 0.8 if $x < 0$			
Input Arguments:  Type  Scalar	Precision SP		How Passe	<u>ed</u>	<u>Units</u>
Output Results: Type Integer	Precision DP		How Pass	<u>ed</u>	<u>Units</u>
Errors Detected: Error # 15		Cause ar too large to ger conversion	1.	Fixup Return either Posmax = X' or Negmax = X'	7FFFFFFF '
Comments:					
	Registers Unsa	fe Across Cal.	L: R4,R5,F	J, E.T.	
Algorithm:				-than	
Second registe routine merges		point registe	r pair is	creared, then	

REPRODUCIBILITY OF THE ORIGINAL PAGE

ROUND	
KUUUND	

Secondary 1	Entry Name:	DCEIL			
Function:	Performs H	AL/S CEILING function	on: Finds the sma	llest integer	
Invoked by	•				
		code for HAL/S cons		<b>n:</b>	
Other	library mod	lules:			
Execution '	Time (micros	seconds): 26.6 if X 36.0 if X			
Input Argu Type Scalar		Precision DP	How Passe F0, F1	<u></u>	<u>Units</u>
Output Res Type Intege		Precision DP	How Passe	<u>:d</u>	<u>Units</u>
Errors Det Erro 15	<u>r #</u>	integer con	large for version.	Fixup Return either Posmax = X' or Negmax = X'	7FFFFFF 8000000
Comments: Algorithm:	Negative an more positiv	egs become less negative. $\alpha$ $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	β	(Cont'd. at h	ıs
Same as 1 Negative are	DROUND, exc guments are	cept positive argume not rounded.		p by almost 1.	

Comments: (Continued)

Registers Unsafe Across Call: R4,R5,F0,F1.

ROUND

Secondary Entry N	Jame: DFLOOR		
	orms HAL/S FLOOR function argument.	on: Finds the largest integ	ger
Invoked by:			
	tted code for HAL/S con where X is a double pre		
Other librar	y modules:		
Execution Time (m	nicroseconds): 27.0 if 36.4 if		
Innut Augumenta.			
Input Arguments: Type	Precision	How Passed	Units
Scalar	DP	FO, F1	-
Output Results:			
Type	Precision	How Passed	Units
Integer	DP	<b>R5</b>	<del></del> . 1 <del></del> 
Errors Detected:			
Error #	<u>Cau</u> Scalar too		<del></del>
	integer con	version. Posma	either: x - X'7FFFFFFF' x = X'80000000'
Comments: Negati	ve arguments become mor	e negative, positive argum	
	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		bottom of page
Algorithm:	FLOOR(α) FLO		
	except argument is ro		
Comments: (Cont'c			

Registers Unsafe Across Call: R4,R5,F0,F1.

ROUND		ROUND			
-------	--	-------	--	--	--

Secondary Entry Name	DROUND			
Function: Convert	s double precision sc	alar to fullword	integer.	
Invoked by:				
	d code for HAL/S cons X is a double precis		<b>m</b> :	
Other library mo	odules:			
Execution Time (micro	oseconds): 33.8			
Input Arguments:  Type Scalar	Precision DP	How Pass FO, F1	<u>ed</u>	Units
Output Results: Type Integer	Precision DP	How Pass R5	<u>ed</u>	<u>Units</u>
Errors Detected: Error #	Cause Scalar too la integer conve	arge for ersion.	Fixup Return either Posmax = X'	7FFFFFFF '
Comments:			or Negmax = X'	
Negative arguments	are converted to the	next more negat	ive integer va	lue;

Negative arguments are converted to the next more negative integer value; positive args to the next greater positive integer value, unless the (Con't on Algorithm:

The argument is checked for negative/not negative. If the argument is negative, the value is rounded down by subtracting just under ½. The resulting value is then checked against MAXNEG(X'C880000000FFFFFF). If within the legal range, the integer part of the scalar is shifted to the second register of the floating point register pair. This remaining integer value is then put in a fixed point register and complemented to leave it in the correct two's-complement fixed point form. If the argument is not negative, the value is rounded up by adding almost ½, and the resulting value is compared to MAXPOS (X'487FFFFFFFFFFFFF). Then, as with negative values, it is shifted to leave the integer part in floating point format and loaded into a fixed point register.

Comments (Con't): original argument is an integer (argument rounded up or down by not quite I before truncating decimal places).

Registers Unsafe Across Call: R4,R5,F0,F1.

Secondary Entry Name: DTOI
Function: Converts double precision scalar to fullword integer.
Invoked by:
X Compiler emitted code for HAL/S construct of the form:
<pre>I = D; where I is a double precision integer, and</pre>
Other library modules:
Execution Time (microseconds): 33.8
Input Arguments:
Type Precision How Passed Units
Scalar DP FO -
Output Results:
Type Precision How Passed Units
Integer DP R5
Errors Detected:
Cause Fixup  15 Scalar too large for integer conversion. Return either:
15 Scalar too large for integer conversion. Return either; POSMAX or
NEGMAX
Comments: DTOI is identical entry point to DROUND.
Registers Unsafe Across Call: R4,R5,F0,F1.

REPRODUCIBILITY OF THE ORIGINAL PAGE 18 POOR

: :

Algorithm:

Same as DROUND.

ROUND.	
TOO MA	

			DIDITAL
Secondary	Entre	Mamo.	DTRUNC
secondar y	THE CT Y	Manie .	

Function: Performs HAL/S TRUNCATE function: Finds the signed value that is the largest integer < absolute value of the argument.

Invoked by:

- X Compiler emitted code for HAL/S construct of the form: TRUNCATE(X), where X is a double precision scalar.
- Other library modules:

Execution Time (microseconds):

Input Arguments:

Type Precision How Passed FO, F1 Scalar

Output Results:

Precision How Passed Type Units Integer DP R5

Errors Detected:

Error # Cause Fixup Scalar too large for 15 Return either: integer conversion. Posmax: X'7FFFFFFF' or Negmax: X'80000000'

Comments: After truncation, negative and positive arguments are closer to 0;

no rounding done before truncation. (Continued at bottom of page)

Algorithm:

DROUND, except argument is not rounded up or down. Same as

Comments: (Continued)

Registers Unsafe Across Call: R4, R5, F0, F1.

ROUND

Secondary Entry Name:	ETOI	•			
Function: Converts sin	gle precision	scalar to fu	ıllword integ	er.	
Invoked by:					
Compiler emitted of I=S where I is scalar.				gle prec	ision
Other library mode	ules:				
Execution Time (micros	econds): 39.0				
Input Arguments:					
Type	Precision		How Passed		Units
Scalar	SP		FO		<del>-</del>
Output Results:					
Type	Precision		How Passed		Units
Integer	<b>DP</b> :		R5		<del>-</del>
Errors Detected: Error #		Cause		Fixup	
		oo large for conversion.			X'7FFFFFFF'
Comments:     ETOI is identical en     Registers Unsafe Ad Algorithm:	ntry point to pross Call: R	ROUND; als		negmax =	x'80000000'
Same as ROUND,					

ROUND	

Secondary Entry Name: FLOO	OR		
Function: Performs HAL/S FLAthe argument.  Invoked by:	OOR function: Returns	largest integer ≤	
Compiler emitted code f	or HAL/S construct of a single precision sca	the form: lar.	
Other library modules:			
Execution Time (microseconds	31.2 if $X \ge 0$ 40.6 if $X \le 0$		
Input Arguments: Type Prec Scalar SI		How Passed FO	Units -
Output Results:  Type Prec		How Passed R5	<u>Units</u>
Errors Detected: Error # 15	<u>Cause</u> Scalar too large for integer conversion.	r Returns either Posmax = X or Negmax = X	'7FFFFFFF'
Algorithm:	cross Call: R4,R5,F0,		en

Second register of floating point register pair is cleared, then

routine merges with DFLOOR.

70	ar i	ND
- 10	UU	1111

Secondary Entry Name:	TRUNC		
Function: Performs HA is the largest inte	L/S TRUNCATE function: eger $\leq$ absolute value of	Returns signed f the argument.	value that
Invoked by:			
Input Arguments:  Type  Scalar	Precision SP	How Passed F0	<u>Units</u>
Output Results: Type Integer	Precision DP	How Passed	<u>Units</u>
Errors Detected: Error # . 15	Cause Scalar too large integer convers	ion. P	Fixup rn either: osmax = X'7FFFFFFF' legmax = X'80000000'
Algorithm:	e Across Call: R4,R5,F of floating point regi	O,F1.	

REPRODUCIEULITY OF THE ORIGINAL PAGE IS POOR

## 5.3.2 Algebraic Routine Description

This subsection presents the detailed descriptions of "Algebraic" routines as defined in Appendix C of the HAL/S Language Specification.

		and the second s	ACOS	<b>.</b>
	HAL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name	e: ACOS Siz	e of Code Area:	102	Hw
Stack Requirement	: <u>24</u> Hw Dat	a CSECT Size:	2	Hw
☐ Intr	insic [	Procedure		
Other Library Mod	ules Referenced:	SQRT		
ENTRY POINT DESCR	IPTIONS			
Primary Entry Name	e:ACOS			
Function: Compu	tes arc-cosine(x) of s	calar argument		
		January and January.		
Invoked by:				
X Compiler emit	tted code for HAL/S co	nstruct of the fo	orm;	
ARCCOS(X), W	where X is a single pre			
	were wars a smidte bre	cision scalar.		
		cision scalar.		
Other Library		cision scalar.		
		cision scalar.		
Other Library	y Modules:			
Other Library	y Modules: icroseconds): .5 <  x	$\begin{vmatrix} \leq 1 : 225.5 \\ 2 \times 10^{-4} <  x  < 1 \end{vmatrix}$	: 132.7	
Other Library	y Modules: icroseconds): .5 <  x		: 132.7 71.5	
Other Library  Execution Time (minimal Input Arguments:  Type	y Modules: icroseconds): .5 <  x	$\begin{vmatrix} \leq 1 : 225.5 \\ 2 \times 10^{-4} <  x  < 1 \end{vmatrix}$	71.5	<u>U</u> n
Other Library  Execution Time (mi	y Modules: icroseconds): .5 <  x 2.44140625  x  < 2.4	$\begin{vmatrix} \leq 1 : 225.5 \\ 2 \times 10^{-4} <  x  < .5 \\ 41406252 \times 10^{-4} \end{aligned}$	71.5	<u>Un</u>
Other Library  Execution Time (mi  Input Arguments:  Type	y Modules:  icroseconds): .5 <  x	$\begin{vmatrix} \leq 1 & 225.5 \\ 2 \times 10^{-4} <  x  < 541406252 \times 10^{-4} & \text{How Pass} \end{vmatrix}$	71.5	<u>Un</u>
Other Library  Execution Time (minimal imput Arguments:  Type Scalar	y Modules:  icroseconds): .5 <  x	$\begin{vmatrix} \leq 1 & 225.5 \\ 2 \times 10^{-4} <  x  < 541406252 \times 10^{-4} & \text{How Pass} \end{vmatrix}$	71.5 sed	
Other Library  Execution Time (mi  Input Arguments:  Type Scalar  Output Results:	y Modules: icroseconds): $.5 <  x $ $2.44140625$ $ x  \le 2.4$ Precision SP	$\begin{vmatrix} \leq 1 & 225.5 \\ 2 \times 10^{-4} <  x  < 541406252 \times 10^{-4} & \text{How Pass} \\ \hline & \text{FO} \end{vmatrix}$	71.5 sed	<u>Un</u>
Other Library  Execution Time (mi  Input Arguments:  Type Scalar  Output Results:  Type Scalar	y Modules:  icroseconds): $.5 <  x $ $2.44140625$ $ x  \le 2.44$ Precision  SP	$\begin{vmatrix} \le 1 : 225.5 \\ 2 \times 10^{-4} <  x  < .5 \end{vmatrix}$ $141406252 \times 10^{-4} : $ How Pass	71.5 sed	<u>Un</u>
Other Library  Execution Time (minimal imput Arguments:  Type Scalar  Output Results:  Type	y Modules:  icroseconds): $.5 <  x $ $2.44140625$ $ x  \le 2.44$ Precision  SP	$\begin{vmatrix} \le 1: 225.5 \\ 2 \times 10^{-4} <  x  < .5 \\ 41406252 \times 10^{-4} : \\ \hline \text{How Pass} \\ \hline \text{F0} \\ \\ \hline \text{How Pass} \\ \hline \text{F0}$	71.5 sed	<u>Un</u>
Other Library  Execution Time (minimal imput Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected:	y Modules:  icroseconds): .5 <  x   2.44140625    x   < 2.4    Precision   SP   Precision   SP   Caus	$\begin{vmatrix} < 1; 225.5 \\ 2 \times 10^{-4} <  x  < .5 \\ 41406252 \times 10^{-4}; \\ \hline How Pass \\ \hline F0 \\ \hline How Pass \\ \hline F0 \\ \hline $	71.5 sed sed Fixup	<u>Un</u> :
Other Library  Execution Time (minus Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected: Error #	y Modules:  icroseconds): .5 <  x   2.44140625    x   < 2.44    Precision   SP   Precision   SP   SP	$\begin{vmatrix} < 1; 225.5 \\ 2 \times 10^{-4} <  x  < .5 \\ 41406252 \times 10^{-4}; \\ \hline How Pass \\ \hline F0 \\ \hline How Pass \\ \hline F0 \\ \hline $	71.5 sed	<u>Un</u> :
Other Library  Execution Time (minus Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected: Error #	y Modules:  icroseconds): .5 <  x   2.44140625    x   < 2.4    Precision   SP   Precision   SP   Caus	$\begin{vmatrix} < 1; 225.5 \\ 2 \times 10^{-4} <  x  < .5 \\ 41406252 \times 10^{-4}; \\ \hline How Pass \\ \hline F0 \\ \hline How Pass \\ \hline F0 \\ \hline $	71.5 sed sed Fixup	<u>Un</u>

### REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

ACO	3
1100.	_

Secondary	Entry	Name:	ASIN	
Decem_1				

Function:

Computes arc-sine of scalar argument.

Invoked by:

- Compiler emitted code for HAL/S construct of the form:

  ARCSIN(X), where X is a single precision scalar.
- Other library modules:

Execution Time (microseconds): 
$$.5 < |x| \le 1$$
: 227.6  
2.441406252 x  $10^{-4} < |x| \le .5$ : 118.4  
 $|x| < 2.441406252$  x  $10^{-4}$ : 57.2

Input Arguments:

Type Precision How Passed Units
Scalar SP F0

Output Results:

Type Precision How Passed Units
Scalar SP F0 radians

Errors Detected:

Error # Cause Fixup

10 argument outside legal range. Return 0.0

Comments

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: The value of X is restricted to  $0 \le X \le 1$  by using the identity  $\arcsin(X)$  -arcsin(X), and further to  $0 \le X \le .5$  by the identity  $\arcsin(X) = \pi/2 - 2$  arcsin( $\sqrt{1-X}$ ). In this range,  $\arcsin(X)$  is computed

as a truncated continued fraction in  $X^2$ , multiplied by X. The form of the approximation is:  $\frac{1}{4}$   $x^3$ 

arcsin(X) 
$$\approx$$
 X +  $\frac{d_1 \times d_1}{c_1 + \times d_2}$   $\frac{d_2}{c_2 + \times d_2}$ 

where the values of the constants are:

 $c_1 = X'C13B446A' = -3.7042025$   $c_2 = X'C11DB034' = -1.8555182$   $c_3 = X'C08143C7' = -0.5049404$ 

		<b>.</b>		AC	OSH	
	HAL	/S-FC LIBRAR	Y ROUTINE DESC	CRIPTION		
	Source Member Name: _	ACOSH	Size of Coo	de Area:3	6	Hw
	Stack Requirement:	20 Hw	Data CSECT	Size: 2		Hw .
-	☐ Intrinsi	<b>c</b>	Proce	dure		
	Other Library Modules	Referenced:	LOG, SQI	RT		
	ENTRY POINT DESCRIPTI	ONS				
	Primary Entry Name:	ACOSH				•
	Computes hype	rbolic arc-c	osine in sing]	e precsion.		
	Invoked by:  X Compiler emitted	code for HA	L/S construct	of the form:		
	ARCCOSH(x), wher	e x is a sin	gle precision	scalar.		
	Other Library Moo	lules:				
	Execution Time (micros		1.6777722E+7 <u>&lt;</u> 1 <u>&lt;</u> x < 1.6777		3	
	Input Arguments: Type	Precision		How Passed		Units
	scalar	SP		FO		
	Output Results:					
	Type	Precision		How Passed		Units
	scalar	SP		FO		
	Errors Detected: Error #		Cause		Fixup	
	59		ARG < 1	Ret	turn 0.0	
	Comments: Registers Unsafe A	cross Call:	FO,F1,F2,F3,F	4,F5.		
	Algorithm:					
	Using the external					
	arccosh(x) :	= $log(x + V)$	x <sup>2</sup> -1)			

			ASINH	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	Source Member Name:ASINH	Size of Code Area:	64	Hw
	Stack Requirement: 20 Hw	Data CSECT Size:	0	Hw
e e e e e e e e e e e e e e e e e e e	☐ Intrinsic	⚠ Procedure		:
	Other Library Modules Referenced:			
<u></u>	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: ASINH			
	Function:			
		arc-sine in single pr	ecision.	
	Invoked by:			
	X Compiler emitted code for HAL ARCSINH(X), where X is a scal		orm:	
		aL.		
	Other Library Modules:			
	Execution Time (microseconds): (Se			
	Execution Time (microseconds): (S	se below)		
	Input Arguments:			
	Type Precision Scalar Sp	How Pass	<u>sed</u>	Units
		<b>F</b> 0		
	Output Results: Type Precision	How Pass		
	Scalar SP	F0	<u>sea</u>	Units
	Errors Detected:			
	Error #	<u>Cause</u>	Fixup	
	none			
	Comments:			
	Registers Unsafe Across (	Call: F0,F1,F2,F3,F4,1	<b>F</b> 5.	
	Algorithm:			
	Using the external SQRT and LOG r			
	$arcsinh(X) = log(X + \sqrt{X^2})$	+1		
	Execution Time: $ X  < 8.8721751E$	4: 31.5		
	$ X  \ge 1.6777722E7$ 8.8721751E-4 <	4: 31.5 : 141.2 x   < 2.1632855E-1: 85.   < 1.6777722E+7: 314.	4	
	2.163255E-1 < X	<pre>&lt; 1.6777722E+7: 314.</pre>	.1	

			ATANH	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		1
i	Source Member Name: ATANH	Size of Code Area:	58	Hw
	Stack Requirement: 18 Hw	Data CSECT Size:	2	Hw
	[ Intrinsic	X Procedure		
	Other Library Modules Referenced:	LOG		
	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: ATANH	A CONTRACTOR OF THE PROPERTY O		
	Function:			
	Computes hyperbolic arc-ta	ingent in single prec	lsion.	
	Invoked by:  X Compiler emitted code for HAL,	S construct of the f	orm:	
	ARCTANH(X), where X is a singl			
	Other Library Modules:			
		x  < 4.113892E-5: 33 .113892E-5 $\leq  x  \leq 1$ .875E-1 $<  x : 228.2$		
	Input Arguments:	How Pas	haz	Units
	Type Precision Scalar SP	FO		
	SCATALLY OF THE STATE OF THE ST			
	Output Results: Type Precision	How Pas	sed	Units
	Scalar SP	FO		-
	Errors Detected:			
	Error #	Cause	Fixup	
	60 argument -1 < X <	coutside range:	Return 0.0	
	Comments:  Registers Unsafe Across Call:	FO,F1,F2,F3,F4,F5.		
	Algorithm:			
	Using the external LOG function,			
	$arctanh(S) = \frac{1}{2} log((1 -$	+ X)/(1 - X).		
	Error #60 is sent if $1 - X = 0$ Which, taken together, are equivalent	0, or if $(1 + X)/(1 - X)$		

<del></del>				DACOS	
		- /	POUMTING PROGREDUITO		-
			ROUTINE DESCRIPTION		
	Source Member Name:	DACOS	Size of Code Area	: 150	Hw
	Stack Requirement: _	18 Hw	Data CSECT Size:	2	Hw
i e	☐ Intrins	sic	X Procedure		
	Other Library Module	es Referenced:	DSQRT		
	ENTRY POINT DESCRIPT	PIONS			
	Primary Entry Name:	DACOS			
	Function: Compute	s ARCCOS(X) in	double precision.		
			an agustudēna lipera saat me Atrika		
		ere X is a doub  Modules:	L/S construct of the ple precision scalar $ x  \le 3.7252907E-7:$ .7252907E-7 < $ x  <$	• 89.1	
	Though Avenuentes		$.5 <  x  \le 1:460.\overline{5}$		cases)
	Input Arguments: Type	Precision	How P	assed	Units
	Scalar	DP	FO		
	Output Results:	Precision	How P		Units
	<u>Type</u>				
	Scalar Errors Detected:	DP	-		radians
	Error #		Cause	Fixup	
	10		ument outside range < X < 1	Return (	0.0
	Comments:				
	Registers Unsafe	Across Call:	FO,F1,F2,F3,F4,F5.		
	Algorithm:				

Computed as  $\pi/2$  - ARCSIN(X)

DASIN Secondary Entry Name:

Function:

Computes ARCSIN(X) in double precision.

Invoked by:

Compiler emitted code for HAL/S construct of the form: ARCSIN(X), where X is a double precision scalar.

Other library modules:

 $|x| \le 3.7252907E-7: 64.1$ Execution Time (microseconds):  $3.725\overline{2907E-7} < |x| \le .5$ : 238.1 .5 <  $|x| \le 1$ : 470.3 (89.5 in odd cases)

Input Arguments:

Type Scalar Precision DP

How Passed F0

Units

Output Results:

Type Scalar Precision

How Passed

Units radians .

ľ.

Errors Detected:

Error # 10

Cause argument outside range -1 < x < 1

Return 0.0

Comments:

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: The value of X is restricted to  $0 \le X \le 1$  by using the identity arcsin(-X) = arcsin(X).

and further to  $0 \le X \le \frac{1}{2}$  by using the identity arcsin(X) = $\pi/2$  -2 arcsin  $\frac{1-X}{2}$ 

Within this range, arcsin(X) is computed as a truncated continued fraction in  $X^2$ , multiplied by X.

ithin this range, 
$$\arcsin(X)$$
 is computed as a truncated continued  $X^2$ , multiplied by X.

The form of the approximation is:
$$\arcsin(X) = X + X^3 \left( C_1 + \frac{d_1}{x^2 + C_2 + d_2} + \frac{d_2}{x^2 + C_3 + d_3} + \frac{d_3}{x^2 + C_4 + d_4} + \frac{d_4}{x^2 + C_5} \right)$$

#### DACOS

### Algorithm (Con't)

where the values of the constants are:

 $C_1 = X'3F180CD96B42A610' = .00587162904063511$ 

d, = X'C07FE600798CBF27' =-.49961647241138661

 $C_2 = X'C1470EC5E7C7075C' = -4.44110670602864049$ 

d<sub>2</sub> = X'C1489A752C6A6B54' =-4.53770940160639666

 $C_3 = X'C13A5496A02A788D' = -3.64565146031194167$ 

d<sub>3</sub> = x'C06B411D9ED01722' =-.41896233680025977

 $C_4 = X'C11BFB2E6EB617AA' = -1.74882357832528117$ 

d<sub>4</sub> = X'BF99119272C87E78' =-.03737027365107758

C<sub>5</sub> = X'C11323D9C96F1661' =-1.19625261960154476

			DACOSH	
e og erekt og sekret Gjerek <b>H</b>	AL/S-FC LIBRARY	ROUTINE DESCRIPTI	ON	
Source Member Name:	DACOSH	Size of Code Are	a: 42	Hw
Stack Requirement:	22 Hw	Data CSECT Size:	2	Hw
☐ Intrin	sic	Procedure		
Other Library Modul	es Referenced:	DLOG, DSQRT		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	DACOSH			
Invoked by:		arc-cosine in doub		
ARCCOSH(X),	where X is a do	uble precision sca	lar.	
Other Library	Modules:			
Execution Time (mice Input Arguments: Type scalar		108869E+7 < X: 332	Passed	<u>Units</u>
Output Results: Type scalar	Precision DP	<u>How</u> F(	Passed	<u>Units</u>
Errors Detected: Error #		Cause	Fixup	
. 59	arg	nument < 1	return (	0.0
Comments: Registers I	Jnsafe Across Ca	11: FO,F1,F2,F3,	F4,F5.	
Algorithm:				
	rnal DSQRT and I			
and the state of t	recosh(x) = log	$(x + \sqrt{x^2-1})$		
			REPRODUCIBI	TATY OF TH

	Ι	DASINH	
HAL/S-FC LIBRARY ROUTIN	NE DESCRIPTION		
	of Code Area:	94	Hw
	CSECT Size:	0	Hw
	Procedure		-
Other Library Modules Referenced: DSQ	RT, DLOG		
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: DASINH			
Function: Computes hyperbolic arc-sine in doub	le precision.		
Invoked by:  KX Compiler emitted code for HAL/S cons  ARCSINH(X), where X is a double pred		<b>m</b> :	
Other Library Modules:			
Execution Time (microseconds): (See b	elow).		
Input Arguments: Type Precision Scalar DP	How Passe	<b>ed</b>	<u>Units</u>
Output Results:  Type Precision Scalar DP	How Passe FO	ed ***	Units -
Errors Detected: Error # Caus		Fixup	
하님, 네트스 이번 골요, 생기의 털어족			
Comments: Registers Unsafe Across Call: F0,F1	,F2,F3,F4,F5.		
Algorithm:			
Using the external DSQRT and DLOG function $\operatorname{arcsinh}(X) = \log (X + \sqrt{X^2 + 1})$	n:		
Execution Time: $ X  \le 1.353860E-8$ : $6.7108864E+7 \le  X  \le 1.353860E-8$ : $1.353860E-8 <  X  \le 6.35E-02 <  X  < 6.7108$	33.6 448.2 25E-2: 185.4 8864E+7: 570.8		

			DATANH	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	Source Member Name: DATANH	Size of Code Area:		Hw
	Stack Requirement: 18 Hw		2	Hw
- 1,000,000	☐ Intrinsic	Procedure		
	Other Library Modules Referenced:	DLOG		
:	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: DATANH			
	Function:			
	Computes hyperbolic arc-t	angent in double pred	cision.	
	Invoked by:  X Compiler emitted code for HAI  ARCTANH(X), where X is a dox			
	Other Library Modules:			
	1.6	x  < 1.0774559E-8: 4  0774559E-8  <  x  < 6  25E-2  <  x : 399.0		
	Input Arguments: Type Precision	How Pa	ssed	Units
	Scalar DP	FO		
	Output Results:  Type Precision	How Pa	ssed	Units
	Scalar DP	FO		
	Errors Detected:			
	Error #	Cause	Fixup	
	60 Argume -1 < X	nt outside range: < 1	Return 0.0	
	Comments:			
	Registers Unsafe Across Call:	F0,F1,F2,F3,F4,F5.		
	Algorithm:			
	Using the external DLOG library fu	nction,		
	$\operatorname{arctanh}(X) = \frac{1}{2}$	$\log ((1 + X)/(1-X)).$		

		DATAN2	<u> </u>
HZ	AL/S-FC LIBRARY ROUTINE	DESCRIPTION	
Source Member Name:	Size of	Code Area: 194	Hw
Stack Requirement:	18 Hw Data CS	ECT Size:	· 26 Hw
<pre>Intrins</pre>	sic 🛛 🗓	Procedure	
Other Library Module	es Referenced: None	2	
ENTRY POINT DESCRIPT	rions	当时40000000000	
Primary Entry Name:	DATAN2		
Function: Computes in	arctan by fraction appro	eximation in the range	e (-T, T)
Compiler emitted ARCTAN2(X,Y), we to sine an	where X and Y are double and cosine respectively or	precision scalar corr the intended arc tar	esponding ngent argur
ARCHAN2(X,Y), we to sine ar	nd cosine respectively of	precision scalar corr the intended arc tar	esponding ngent argu
ARCTAN2(X,Y), we to sine an	nd cosine respectively of	precision scalar corr the intended arc tar	responding ngent argu
ARCHAN2(X,Y), we to sine an Other Library M	nd cosine respectively of	precision scalar corr the intended arc tar	responding ngent argu
ARCTAN2(X,Y), we to sine an Other Library Market Time (microscopic Time (microscopic Time Type)	nd cosine respectively of Modules: coseconds): 248.4  Precision	the intended arc tar	ngent argu
ARCTAN2(X,Y), we to sine an Other Library Modern Li	nd cosine respectively of Modules: coseconds): 248.4 <pre>Precision</pre> <pre>DP</pre>	the intended arc tar  How Passed F0	ngent argu
ARCTAN2(X,Y), we to sine an Other Library Market Time (microscopic Time (microscopic Time Type)	nd cosine respectively of Modules: coseconds): 248.4  Precision	the intended arc tar	ngent argu
ARCTAN2(X,Y), we to sine and to sine (microscapidate).  Execution Time (microscapidate).  Input Arguments:  Type Scalar (sin) Scalar (cos)  Output Results:	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP	the intended arc tar  How Passed F0	ngent argu
ARCTAN2(X,Y), we to sine and to sine (microsoft). Type  Scalar (sin) Scalar (cos)  Output Results:  Type	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP Precision	the intended arc tar  How Passed F0	responding ngent argu <u>Ur</u>
ARCTAN2(X,Y), we to sine and to sine (microscapidate).  Execution Time (microscapidate).  Input Arguments:  Type Scalar (sin) Scalar (cos)  Output Results:	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP	How Passed F0 F2	ngent argu
ARCTAN2(X,Y), we to sine and to sine (microsoft). Type  Scalar (sin) Scalar (cos)  Output Results:  Type	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP Precision	How Passed F0 F2 How Passed	ngent argu
ARCTAN2(X,Y), we to sine and to sine (microsoft). Type Scalar (sin) Scalar (cos)  Output Results:  Type Scalar	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP Precision	How Passed F0 F2 How Passed	ungent argu Un Rac
ARCTAN2(X,Y), we to sine an consider the sine and the consider the consideration of the consi	nd cosine respectively of Modules:  coseconds): 248.4  Precision DP DP Precision DP	How Passed F0 F2  How Passed F0	ugent argu Vi Rad
ARCTAN2(X,Y), we to sine an to sine an to sine an other Library M.  Execution Time (microsocial form)  Input Arguments:  Type Scalar (sin) Scalar (cos)  Output Results:  Type Scalar  Errors Detected: Error #	Modules:  Coseconds): 248.4  Precision DP DP Precision DP Cause	How Passed F0 F2  How Passed F0 F0 F1	ungent argument argum

Again, 
$$Z = \frac{|\sin x|}{|\cos x|}$$
. Special cases - (1) If  $\cos x < 0$  and  $Z < (16)^{-14}$ , return  $\pm \pi$ . (2)  $\sin x = \cos x = 0$ , signal error and return 0. (3)  $\sin x \neq 0$ ,  $\cos x = 0$ , return  $\pm \pi/2$ . (4)  $\sin x \neq 0$ ,  $\cos x \neq 0$ , but  $Z > (16)^{14}$ , return  $\pm \pi/2$ . (5) If  $Z < (16)^{-7}$ , return  $Z$ .

## REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

DATAN2

DATAN2

Algorithm (Con't)

The fractional approximation after reduction of Z to < tan 15° is:

$$Tan^{-1}(Z) = Z + Z \cdot Z^2 \cdot F$$
, where

 $F = C1 + C2/(z^2 + C3 + C4/(z^2 + C5 + (C6/(z^2 + C7)))).$ 

C1 = X'BF1E31FF1784B965' (-0.7371899082768562E-2)

C2 = X'COACDB34COD1B35D' (-0.6752198191404210)

C3 = X'412B7CE45AF5C165' (0.2717991214096480E+1)

C4 = X'CllA8F923B178C78' (-0.1660051565960002E+1)

CS = X'412AB4FD5D433FF6' (0.2669186939532663E+1)

C6 = X'C02298BB68CFD869' (-0.1351430064094942)

C7 = X'41154CEE8B7DCA99' (0.1331282181443987E+1)

As in EATAN2, the intermediate result is adjusted to the proper section in the first quadrant, as follows:

(original)  $Z \le \tan 15^{\circ} \rightarrow + 0$ 

 $tan 15^{\circ} < z \le 1 \rightarrow + \pi/6$ 

 $1/Z < \tan 15^{\circ}$   $\rightarrow$   $(-\pi/2 + 1)$  then -1 (to preserve signif. bits)

tan  $15^{\circ} < 1/2 < 1 \rightarrow (-\pi/3 + 1)$  then -1 (to preserve signif. bits)

The resulting angle is adjusted to the proper quadrant as in EATAN2 (according to sign of  $\sin x$  and  $\cos x$ ).

DATAN2		

Secondary Entry Name: DATAN
Function: Computes are tangent by fractional approximation in the range $(-\pi/2, +\pi/2)$ in double precision.
Invoked by:
Compiler emitted code for HAL/S construct of the form: ARCTAN(X), where X is a double precision scalar.
Other library modules:
Execution Time (microseconds): 237.3
Input Arguments:  Type Precision How Passed Units Scalar DP FU
Output Results:  Type Precision How Passed Units Scalar DP F0 Radian
Errors Detected: Error # Cause Fixup
Comments: Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm:

Same as ARCTAN, but see DATAN2 for changes in values of DP constants and  ${\rm TAN}^{-1}$  formula.

			DEXP	
на	L/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	DEXP	Size of Code Area:	154	Hw
Stack Requirement: _	18 Hw	Data CSECT Size:	66	Hw
☐ Intrins	ic	Procedure		
Other Library Module	s Referenced:	None		
ENTRY POINT DESCRIPT	ONS			
Primary Entry Name:	DEXP	and the state of t		
Function  Computes e in de	ouble precision			
Invoked by:  A Compiler emitte	d code for HAI	yS construct of the fo	אוינכ:	
EXP(X), whe	re X is a doubl	e precision scalar.		
X Other Library M	iodules:			
	DPWRD DSINH,	DTANH		
Execution Time (micr	oseconds): 29	0.5		
Input Arguments: Type	Precision	How Pass	sed	Units
Scalar	DP	FO		
Output Results: Type	Precision	How Pas	sed	Units
Scalar	DP	FO		
Errors Detected:				
Error #		Cause	Fixup	
		ent outside range .74.6731	Return maximufloating poin	
Comments:				
Gives underflow if Registers Unsafe		all - no error number.		
Algorithm: First, of part and first hexad part of X by LOG2H, rounded up. This is value for R', by expenses the part of X	decompose X as decimal place of which is a single done in 80-bit pressing log2 =	P'log2 + R', where P' of the result of dividingle precision approximate precision in order to L LOG2H + LOG2L, where sign as X, and   R'  n	ing the high-c mation to log2 to yield a tru LOG2L is a c	order 2, 1e 56-bit double
log2				
1692				

## REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

DEXP

DEXP

Algorithm (Con't)

Now, if R' > 0, subtract  $\frac{\log 2}{16}$  from it until it becomes  $\leq$  0, each time adding  $\frac{1}{16}$  to P'. If R'  $\leq -\frac{\log 2}{16}$ , add  $\frac{\log 2}{16}$  to it until it becomes  $> -\frac{\log 2}{16}$ , each time subtracting  $\frac{1}{16}$  from P'.

At the end of this, we have

 $X = P \cdot log2 + R$ , P an integral multiple of 1/16, and  $-\frac{log2}{16} < R \le 0$ .

Represent P as  $4A - B - \frac{C}{16}$ , where A, B, and C are integers,  $0 \le B \le 3$ , 0 < C < 15. Then:

$$e^{X} = 16^{A} \cdot 2^{-B} \cdot 2^{-\frac{C}{16}} \cdot e^{R}$$

To calculate this, we compute e<sup>R</sup> with a polynomial approximation of the form:

$$e^{r} = 1 + c_{1}r + c_{2}r^{2} + c_{3}r^{3} + c_{r}r^{4} + c_{5}r^{5} + c_{6}r^{6}$$

where the values of the constants are:

 $c_3 = X'402AAAAAA794AA99' = .1666666659481656$ 

 $c_4 = X'3FAAAA9D6AC1D734' = .0416666173078875$ 

 $c_5 = X'3F2220559A15E158' = .00833161772003906$ 

 $c_6 = X'3E591893' = .001359497$ 

 $(\bar{\phantom{a}})$ 

Then, 2 is computed by table lookup, 2 by shifting, and 16 by adding A to the exponent of the answer.

		· · · · · · · · · · · · · · · · · · ·	DLOG	-
H.A	AL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	DLOG	Size of Code Area:	140	Hw
Stack Requirement: _	22 Hw	Data CSECT Size:	2	Hw
☐ Intrins	sic	☑ Procedure		
Other Library Module	es Referenced:	None		
ENTRY POINT DESCRIPT	rions			
Primary Entry Name:	DLOG			
Function: Computes log(X) in	double precisi	on.		
		/S construct of the forecision scalar.	orm;	
X Other Library M	DPWRD, DASINH	, DATANH, DACOSH		
Execution Time (micr	roseconds): 28	82.2		
Input Arguments:				
Type	Precision DP	How Pass F0	i <b>ed</b> 1.41	Units
Scalar				
Scalar Output Results: Type	Precision DP	How Pass	ed	<u>Units</u>
Scalar Output Results: Type Scalar	Precision DP	How Pass	ed	<u>Units</u>
Scalar Output Results: Type	DP		Fixup  If X < 0 r  if X=0, re	return log(
Scalar  Output Results:     Type     Scalar  Errors Detected:     Error #     7  Comments:	DP argument	F0	Fixup  If X < 0 r  if X=0, re	return log(
Scalar  Output Results:	argument  a Across Call: $X = 16^P \cdot 2^{-Q}$	F0  Cause  outside range X > 0.	Fixup  If X < 0 r  if X=0, re  negative f  number.  P, Q are int	eturn log(  turn maximu loating poi

DLOG

Algorithm (Con't)

 $\log((1+Z)/(1-Z))$  is computed by an approximation of the form:

$$w + c_1 w^3 \left( w^2 + c_2 + \frac{c_3}{w^2 + c_4 + c_5} \right)$$

where W = 2Z, and the values of the constants are:

 $C_1 = X'3DDABB6C9F18C6DD' = 0.2085992109128247E-3$ 

 $C_2 = X'422FC604E13C20FE' = 0.4777351196020117E+2$ 

 $C_3 = x'C38E5AlC55CEBlC4' = -0.2277631917769813E+4$ 

 $C_4 = X'Cl6F2A64DDFCClFD' = -6.947850100648906$ 

 $C_5 = X'Cl2A017578F548Dl' = -2.625356171124214$ 

 $C_6 = X'C158FA4E0E40C0A5' = -5.561109595943017$ 

			DPWRD	<del>/** id = 4,***</del>
HAL	/S-FC LIBRARY	ROUTINE DESCRIPTION	<b>ON</b>	
Source Member Name: _	DPWRD	Size of Code Area	ı: <u>38</u>	Hw .
Stack Requirement:	22 <sub>Hw</sub>	Data CSECT Size:	4	Hw
☐ Intrinsi		Procedure		
Other Library Modules	Referenced:	DLOG, DEXP, DSQR	T	alest the second se
ENTRY POINT DESCRIPTI	ONS		- 11 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	
Primary Entry Name:	DPWRD			
Function:				
Performs exponentiati	on of double p	orecision scalar to	double preci	ision power
			eria. Territoria	
Invoked by:				
X Compiler emitted				
X**Y, where X and	Y are scalars	and at least X or	Y is double p	recision.
			***	
	dules:			
Other Library Mo	dules:			
U Other Library Mo	dules:			
Execution Time (micro		Y = .5: 238.4		
	seconds): If	Y = .5: 238.4 Y ≠ .5: 635.6		
Execution Time (micro	seconds): If	Y = .5: 238.4 Y ≠ .5: 635.6		
Execution Time (micro	seconds): If If	Y ≠ .5: 635.6		
Execution Time (micro Input Arguments: Type	seconds): If If <u>Precision</u>	Y ≠ .5: 635.6 <u>How P</u>	assed	Ųni
Execution Time (microstant) Input Arguments: Type Scalar (base)	seconds): If If  Precision DP	Y ≠ .5: 635.6 <u>How P</u> F(		<u>Uni</u>
Execution Time (micro Input Arguments: Type	seconds): If If <u>Precision</u>	Y ≠ .5: 635.6 <u>How P</u>		Uni
Execution Time (microstrope Input Arguments:  Type Scalar (base) Scalar (exponent)	seconds): If If  Precision DP	Y ≠ .5: 635.6 <u>How P</u> F(		Uni
Execution Time (microstructure)  Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:	seconds): If If <u>Precision</u> DP DP	Y ≠ .5: 635.6  How P  F(		
Execution Time (microstructure)  Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results: Type	seconds): If  Precision  DP  DP  Precision	Y ≠ .5: 635.6  How P  F( F2	assed	
Execution Time (microstructure)  Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:	seconds): If If <u>Precision</u> DP DP	Y ≠ .5: 635.6  How P  F(	assed	
Execution Time (microstructure)  Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results: Type	seconds): If  Precision  DP  DP  Precision	Y ≠ .5: 635.6  How P  F( F2	assed	
Execution Time (microstrope) Input Arguments: Type Scalar (base) Scalar (exponent) Output Results: Type Scalar	seconds): If  Precision  DP  DP  Precision	Y ≠ .5: 635.6  How P F( F2  How P	assed	Uni
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:  Type Scalar  Errors Detected:	seconds): If  If  Precision DP DP Precision DP	Y ≠ .5: 635.6  How P F( F2  How P F( Cause	assed Fixu	<u>Uni</u>
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:  Type Scalar  Errors Detected: Error #	Precision DP DP Precision DP DP	Y ≠ .5: 635.6  How P F( F2  How P F( Cause	assed Fixu	Uni p 0
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:  Type Scalar  Errors Detected: Error #	seconds): If  If  Precision DP DP Precision DP	Y ≠ .5: 635.6  How P F( F2  How P F( Cause	assed Fixu	Ō
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:  Type Scalar  Errors Detected: Error #	Precision DP DP Precision DP DP	Y ≠ .5: 635.6  How P F( F2  How P F( Cause	assed Fixu	Uni p 0
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4 24  Comments:	seconds): If  Precision DP DP Precision DP base=0; expo	Y ≠ .5: 635.6  How P FC  How P FC  Cause nent ≤ 0	assed Fixu	Uni p 0
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results:  Type Scalar  Errors Detected: Error # 4 24  Comments: Registers Unsafe A	seconds): If  If  Precision DP DP  Precision DP  base=0; expo base < 0	Y ≠ .5: 635.6  How P F( F2  How P F(  Cause nent ≤ 0	Fixu Return 0, use   base	Uni p 0
Execution Time (microstate Input Arguments:  Type Scalar (base) Scalar (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4 24  Comments:	seconds): If  If  Precision DP DP  Precision DP  base=0; expo base < 0	Y ≠ .5: 635.6  How P F( F2  How P F(  Cause nent ≤ 0	Fixu Return 0, use   base	<u>Uni</u> <u>p</u>



		<u>D</u>	PWRI	
HAL/S-	FC LIBRARY ROUTIN	E DESCRIPTION		
Source Member Name:	DPWRI Size	of Code Area:	40	Hw
Stack Requirement:	18 Hw Data	CSECT Size:	2	Hw
☐ Intrinsic		Procedure		
Other Library Modules Re	ferenced: No	one		
ENTRY POINT DESCRIPTIONS				
Primary Entry Name:	DPWRI			
Function:				
Exponentiation of a dou	ble precision sca	lar to a fullword	integer pow	er.
Invoked by:  Compiler emitted continue X**I, where X is a	ode for HAL/S condouble precision	struct of the for scalar; I is a do	m: ouble precis	ion in
Other Library Modul	lac•			
Other Pintary Mode.				
Input Arguments:				TI N
<u>Type</u>	conds): (See ne	xt page)  How Passe F0	<u>a</u>	<u>Un</u> :
Input Arguments:	<b>Precision</b>	<u> How Passe</u>	<u>a</u>	<u>Uni</u>
Input Arguments:  Type Scalar (base) Integer (exponent)	Precision DP	How Passe FO	<u>d</u>	<u>Uni</u>
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type	Precision DP DP DP	How Passe F0 R6 How Passe		
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results:	Precision DP DP	How Passe FO R6		<u>Uni</u>
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected:	Precision DP DP Precision DP	How Passe F0 R6  How Passe F0	<b>∄</b>	
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error #	Precision DP DP Precision DP	How Passe F0 R6  How Passe F0	d Fixup	<u>Un.</u>
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected:	Precision DP DP Precision DP	How Passe F0 R6  How Passe F0	<b>∄</b>	<b>Un</b> :
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error #	Precision DP DP Precision DP	How Passe F0 R6  How Passe F0	d Fixup	<u>Un</u>
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4	Precision DP DP Precision DP  Caus	How Passe F0 R6  How Passe F0  se power < 0	d Fixup	<b>Un</b> :
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4  Comments: Registers Unsafe Ac	Precision DP DP Precision DP  Caus Zero raised to	How Passe F0 R6  How Passe F0  se power < 0	Fixup Return (	<b>Un</b> :
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4  Comments: Registers Unsafe Act Algorithm: If I is the	Precision DP DP Precision DP  Caus Zero raised to	How Passe FO R6  How Passe FO  se power < 0  ,F2,F3.	Fixup Return (	<b>Un</b> :
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4  Comments: Registers Unsafe Ac Algorithm: If I is the	Precision DP DP  Precision DP  Caus Zero raised to ross Call: F0,F1 a fullword exponent = $\sum_{i=0}^{32} e_i 2^i$ , whe	How Passe F0 R6  How Passe F0  ee power < 0  F2,F3.  Tt, D the base, where e = 0 or 1.	Fixup Return (	<b>U</b> n.
Input Arguments:  Type Scalar (base) Integer (exponent)  Output Results: Type Scalar  Errors Detected: Error # 4  Comments: Registers Unsafe Ac Algorithm: If I is the	Precision DP DP  Precision DP  Caus Zero raised to ross Call: F0,F1 a fullword exponent = $\sum_{i=0}^{32} e_i 2^i$ , whe	How Passe F0 R6  How Passe F0  ee power < 0  F2,F3.  Tt, D the base, where e = 0 or 1.	Fixup Return (	<u>Un</u>

**DPWRI** 

DPWRI

Execution Time (microseconds):

If exponent  $\geq$  0: 40.8 + (n-1) 23.2 + 13.0mIf exponent  $\leq$  0: 64.0 + (n-1) 23.2 + 13.0mwhere m = number of 1's in binary representation of exponent.

n = number of significant digits in binary representation
 of |exponent|.

Algorithm (Con't)

To compute  $\pi_{e_i=1}^{D^2}$ , it is only necessary to compute successively  $D^2 = D$ ,  $D^2$ ,  $D^4$ ,  $D^8$ , ..., and multiply the result by  $D^2$  whenever the i-th bit of the exponent is 1. This is determined by shifting bits one by one out of the exponent, and testing each one for a value of one. The loop terminates when the remaining part of the exponent is zero. Operations are done on absolute value of exponent. If exponent was negative, the reciprocal of the result is taken as the final result.

DPWRI
DPWKI

Secondary Entry Name:	DPWRH		
Function: Exponentiation of a	double precisi	on scalar to a halfword inte	eger power.
Invoked by:			
Compiler emitted	code for HAL/S	construct of the form:	
X**I, where X is	a double preci	sion scalar; I is a single p	precision integer
Other library mod	ules:		
Execution Time (micros		e as DPWRI except constants exponent > 0: 41.4 exponent < 0: 64.6	are
Input Arguments:			
Type	Precision	How Passed	Units
Scalar (base)	DP	FO	
Integer (exponent)	SP	<b>R6</b>	
Output Results:			
Type	Precision	How Passed	Units
Scalar	DP	FO	
Errors Detected:			
Error #		<u>Cause</u> <u>F</u>	ixup
. 4	Zero raised t	o power < 0 Re	eturn 0
		in the wife of the engineering of the co	

### Comments:

Registers Unsafe Across Call: F0,F1,F2,F3.

Algorithm: The halfword exponent is shifted right to convert it to a fullword, then the DPWRI algorithm is used.

			DSIN	
HAL	/s-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name: _	DSIN	Size of Code Area:	102	Hw
	20 Hw	Data CSECT Size:	62	Hw
Stack Requirement:		v procedure		
☐ Intrinsi				
Other Library Modules	Referenced:	Notice		
ENTRY POINT DESCRIPTI	IONS			
Primary Entry Name:	DSIN			
Function: Computes sine	(X) in double	precision.		
Invoked by:  X Compiler emitted SIN(X), where X	d code for HA	L/S construct of the i	Form:	
Other Library M	odules:			
Execution Time (micr	oseconds):	267.0		
Input Arguments: <u>Type</u> Scalar	Precision DP	How Pa	ssed	<u>Units</u> radi <b>a</b> ns
Output Results:	Precision	How Pa	ssed	<u>Units</u>
<u>Type</u> Scalar	DP	FO		
Errors Detected:				
Error # 8	Argumen  x  < π	Cause t outside range: .2 <sup>50</sup>	Return $\frac{\text{Fixup}}{2}$	
Comments:		되는 그를 하는 것이 되었다. 그룹을 하는 것이 잘 말았다.		
Registers Unsaf	e Across Call	: FO,F1,F2,F3,F4,F5.		
		ere Q is an integer an		
is desired.		d and $X < 0$ , and add 2		
Since sin(-x) = sin	(+x), and co	$s(x) = \sin(\frac{\pi}{2} + x)$ , this	reduces the	problem
to computing sin(x)				

DSIN

Algorithm (Con't)

Since Q has been adjusted, it is only necessary to compute  $\sin(\frac{\pi}{4}(Q+R))$ . If  $Q'=Q \mod 8$ , then this is equal to  $\sin(\frac{\pi}{4}(Q'+R))$ . The eight cases of this yeild; through simple trigonometric identities:

Q'=0: 
$$\sin(x) = \sin(R \cdot \frac{\pi}{4})$$

Q'=1: 
$$\sin(x) = \cos((1-R) \cdot \frac{\pi}{4})$$

$$Q'=2$$
:  $\sin(\mathbf{x}) = \cos(\mathbf{R} \cdot \frac{\pi}{4})$ 

Q'=3: 
$$\sin(x) = \sin((1-R) \cdot \frac{\pi}{4})$$

$$Q'=4$$
:  $\sin(x) = -\sin(R \cdot \frac{\pi}{4})$ 

$$Q'=5$$
:  $\sin(x) = -\cos((1-R) \cdot \frac{\pi}{4})$ 

Q'=6: 
$$\sin(x) = -\cos(R \cdot \frac{\pi}{4})$$

Q'=7: 
$$\sin(x) = -\sin((1-R) \cdot \frac{\pi}{4})$$

Thus, if we let R'=R in octants 0, 2, 4, 6, and R'=1-R in octants 1, 3, 5, 7. We need only compute  $\sin(R' \cdot \frac{\pi}{4})$ 

in octants 0, 3, 4, 7, and  $cos(R' \cdot \frac{\pi}{4})$  in 1, 2, 5, 6, and take the negative value in octants 4, 5, 6, 7.

 $\sin(R' \cdot \frac{\pi}{4})$  and  $\cos(R' \cdot \frac{\pi}{4})$  are computed by polynomial approximations.

The form of the polynomial approximation for sine is:

$$\sin(R' \cdot \frac{\pi}{4}) = R'(C_0 + C_1R'^2 + C_2R'^4 + C_3R'^6 + C_4R'^8 + C_5R'^{10} + C_6R'^{12})$$

where the values of the constants are:

 $C_0 = X'40C90FDAA22168C2' = .78539816339744831$ 

 $C_1 = X'C014ABBCE625BE41' = -.080745512188280536$ 

 $C_2 = X'3EA335E33BAC3FBD' = 2.4903945701888438E-3$ 

 $C_3 = X'BD265A599C5CB632' = -3.6576204158913872E-5$ 

## REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

DSIN

DSIN

Algorithm (Con't)

 $C_4 = X'3B541E0BF684B527' = 3.1336162254333759E-7$ 

 $C_5 = X'B978C01C6BEF8CB3' = -1.7571500746935669E-9$ 

 $C_6 = x'3778FCE0E5AD1685' = 6.8773605709403589E-12$ 

The form of the polynomial approximation for cosine is:

$$\cos(R' \cdot \frac{\pi}{4}) = 1 + C_1 R'^2 + C_2 \cdot R'^4 + C_3 \cdot R'^6 + C_4 \cdot R'^8 + C_5 \cdot R'^{10} + C_6 \cdot R'^{12} + C_7 \cdot R'^{14}$$

where the values of the constants are:

 $C_1 = X'C04EF4F326F91777' = -.30842513753404242$ 

 $C_2 = X'3F40F07C20606AB1' = 1.5854344243815420E-2$ 

 $C_3 = X'BE155D3C7E3C90F8' = -3.2599188692673765E-4$ 

 $C_4 = X'3C3C3EAOD06ABC29' = 3.5908604460279520E-6$ 

 $C_5 = X'BA69B47B1E41AEF6' = -2.4611364033652271E-8$ 

 $c_6 = x'387E731045017594' = 1.1500512028186245E-10$ 

 $C_7 = X'B66C992E84B6AA37' = -3.8581890061323055E-13$ 

DOTAL
DSIN
~~~

Secondary Entry Name:	DCOS		
Function: Computes cosine(x)	in double precision.		
Invoked by:			
X Compiler emitted	d code for HAL/S cons	truct of the form:	
	is a double precision		
Other library mo			
Execution Time (micro	oseconds): $-\pi \leq x \leq$	π: 261.8	
Input Arguments:	X > π or	x < -π: 264.2	
Type	Precision	How Passed	Units
Scalar	DP	FO	radians
Output Results:  Type  Scalar	Precision DP	How Passed FO	<u>Units</u>
Errors Detected: Error #	Cause		<b>ixup</b>
8	Argument outside  x  < π.250	en e	$\frac{\sqrt{2}}{2}$
Comments:			
Registers Unsafe	Across Call: F0,F1,F	2,F3,F4,F5.	
Algorithm: See DSIN algorithm.			

			DSI	NH
HAL,	/s-FC LIBRARY	ROUTINE DESC	RIPTION	
Source Member Name: _	DSINH	Size of Code	Area: 130	Hw
Stack Requirement:	22 Hw	Data CSECT S	Size:	Hw
☐ Intrinsi	C		ure	
Other Library Modules	Referenced:	DEXP		
ENTRY POINT DESCRIPTION	ons			
Primary Entry Name:		ble precision		
Invoked by:  X Compiler emitted	code for HAL	S construct	of the form:	
SINH(X), where X	is a double p	precision scal	lar.	
Other Library Mo		.81374E-1 <	d < 1 752ccn	
Execution Time (micro	2.	x   < 2.063017	X  < 8.8137	4E-01: 196.7
Input Arguments: <u>Type</u> Scalar	Precision DP		How Passed F0	Units
Output Results: Type	Precision		How Passed	Units
Scalar	DP		FO	
Errors Detected:				
Error #	Argument out: $ x  \le 175.366$			Fixup maximum positive ng point number.
Comments:				
Registers Unsafe Ac	ross Call: FO,	F1,F2,F3,F4,E	5.	
Algorithm: If $ X  <  X  < .881374$ . Then	1.626459E-10 sinh(x) is			
The form of the polyr	nomial is:			
$\sinh(x) = x + c_1 x^3 +$	$c_2 x^5 + c_3 x^7 +$	$c_4^{2}x^9 + c_5^{2}x^{11}$	$+ c_6 x^{13}$	
where the values of t	the con <b>st</b> ants	are:		

DSINH

Algorithm (con't)

 $C_1 = X'402AAAAAAAAAAAA' = 0.166666666666653$ 

 $C_2 = X'3F2222222222BACE' = 0.83333333333367232E-2$ 

 $C_3 = X'3DD00D00CB06A6F5' = 1.984126981270711E-4$ 

 $C_4 = X'3C2E3BC881345D91' = 2.755733025610683E-6$ 

 $C_5 = X'3A6B96B8975A1636' = 2.504995887597646E-8$ 

 $C_6 = X'38B2D4C184418A97' = 1.626459177981471E-10$ 

Otherwise, sinh(|x|) or cosh(|x|) is calculated using EXP. The number V, equal to 0.4995050, is introduced to control rounding errors and the formula is as follows:

$$\sinh(x) = \frac{1}{2v} (e^{(x+\log v)} - \frac{v^2}{e^{(x+\log v)}})$$

$$\cosh(x) = \frac{1}{2v} (e^{(x+\log v)} - \frac{v^2}{(x+\log v)})$$

The identities sinh(-x) = -sinh(x) and cosh(-x) = cosh(x) are used to recover sinh(x) and cosh(x) from sinh(|x|) and cosh(|x|).

Secondary Entry Name	DCOSH DCOSH		
Function: Computes hyper	rbolic cosine in doub	le precision.	
Invoked by:			
	ed code for HAL/S core X is a double preci		
Other library	modules:		
Execution Time (mic	roseconds): $ x  \leq 1$ ,	75366E+2: 422.6	
Input Arguments:			
Type	Precision	How Passed	Units
Scalar	DP	FO FO	<u>onits</u>
Output Results: Type	Precision	How Passed	The date
Scalar	DP	F0	<u>Units</u>
Errors Detected:			
Error #	Cav		ĸup
	Argument outs $ X  \leq 175.366$		aximum positive point number
Comments:	rs Unsafe Across Call	.: F0,F1,F2,F3,F4,F5.	
Algorithm:			
See DSINH algorithm.	om Kolton villigante i tropog. Morato Villia variante i tropog		

		<u>D</u>	SORT	
HA	L/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	DSQRT	Size of Code Area: _	70	Hw
Stack Requirement:		· · · · · · · · · · · · · · · · · · ·	2	Hw Hw
☐ Intrins		Procedure		
Other Library Module		None		
ENTRY POINT DESCRIPT	TIONS			
Primary Entry Name:	DSQRT			
Function:				
	uare root in do	uble precision.		
Invoked by:	un signi est mar	/s construct of the f	orm;	
X Compiler emitte	ed code for HAL	/S construct of the f		
SQRT(X), where	X is a doubte b	recision scalar.		
Input Arguments: <u>Type</u> Scalar	Precision DP	How Pas	ssed	<u>Uni</u>
Output Results:				•••
Type	Precision	How Pa	ssed	<u>Un:</u>
Scalar	DP	. 14 (14 )		
Errors Detected:			Fixup	
Error #	Avens	<u>Cause</u> ment outside of range		qrt( X )
Comments:				
Registers U		ill: F0,F1,F2,F3,F4,F		
Algorithm: Let X	$= 16^{2P+Q} \cdot M, w$	here P, Q are integer	s, $Q = 0$ or $J$	, and
$\frac{1}{16} \leq M \leq 1$ . Then				
	$= 16^{P+Q}$ .	M*16 <sup>-Q</sup> .		
For a first approxi				
<b>У</b>	= A*(M+B) · 16	P 4 <sup>Q+1</sup>		

# REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

DSQRT

DSQRT

Algorithm (Con't)

where the values of the constants are:

A = X'40385F07' = .22020005

B = X'40423A2A' = .25870006

This calculation is carried out with the characteristic of A increased by 8 and the others decreased by 8, in order to store the value of

$$(B \cdot 16^{P+Q}) \cdot 16^{-8}$$

for later use.

Then, two passes of the Newton-Raphson iteration are performed in single precision. The form of the iteration is:

$$y_{k+1} = \frac{1}{2} (\frac{y_3}{y_3} - y_3) + (B \cdot 16^{P+Q-8}) - (B \cdot 16^{P+Q-8}) + y_3$$

single precision

This is done to truncate excess digits of  $\frac{x}{y_3}$  -  $y_3$ , which is 0, and is less than  $16^{P+Q-8}$  in absolute value.

				DTAN	inta <del>=</del> jouenaum ni
		HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	Source Member Na	me: DTAN	Size of Code Area:	164	IJ
	Pource Member Ma	uic. Paris		164	Hw
	Stack Requiremen	t: 30 Hw	Data CSECT Size:	4.	Hw
	☐ Int	rinsic	M Procedure		
	e andre in the contract of the	3	N <b>o</b> ne		
	Other Library Mo	dules Referenced:			
]	ENTRY POINT DESC	RIPTIONS			ja järasillar
	Primary Entry Na	me: DTAN			
		me: DIAW			
I	Function: Computes tar	ngent in double pre	cision.		
1	Invoked by:				
		itted code for HAL, ere X is a double p	/S construct of the for	orm;	
	TYM (V) MIIG	ere v te a domite b	recraton acaidi.		
	Other Libra	www. Modulos			
	C Ocuer prora	TA MOUNTED:			
1	Execution Time (	microseconds): 30	02.2		
			02.2		
	Input Arguments:				
			02.2  How Pass F0	sed	/
	Input Arguments: Type	Precision	How Pass	sed	Unit: radia
	Input Arguments: <u>Type</u> Scalar	Precision	How Pass	sed	/
	Input Arguments: Type	Precision	How Pass		/
	Input Arguments:  Type Scalar  Output Results:	Precision DP	How Pass F0		radia
	Input Arguments:  Type Scalar  Output Results:  Type Scalar	Precision DP	How Pass F0		radia
	Input Arguments:  Type Scalar  Output Results:  Type Scalar  Scalar  Errors Detected:	Precision DP  Precision DP	How Pass F0 How Pass F0 Cause	sed	radia
	Input Arguments:  Type Scalar  Output Results:  Type Scalar	Precision DP  Precision DP  Argument outside	How Pass F0  How Pass F0  Cause range $ X  < \pi.2^{50}$		radia
	Input Arguments:	Precision DP  Precision DP  Argument outside rangument too near	$\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{Cause}}{\text{range } \mathbf{x} } < \pi.2^{50}$ a singularity of	Fixup Return 1 Return max	radia Unit
	Input Arguments:  Type Scalar  Output Results: Type Scalar  Errors Detected: Error #  11 12	Precision DP  Precision DP  Argument outside	$\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{Cause}}{\text{range } \mathbf{x} } < \pi.2^{50}$ a singularity of	Fixup Return 1	radia Unit
	Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected:  Error # 11 12  Comments:	Precision DP  Precision DP  Argument outside rangument too near the tangent func	$\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{How Pass}}{\text{F0}}$ $\frac{\text{Cause}}{\text{range } \mathbf{x} } < \pi.2^{50}$ a singularity of ction	Fixup Return 1 Return max floating	radia <u>Unit</u> imum posi  point num
	Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected:  Error # 11 12  Comments: Error gets v	Precision DP  Precision DP  Argument outside rargument too near the tangent function of the following states of the precision of the tangent function of the precision of the tangent function of the precision of	How Pass  F0  How Pass  F0  Cause $\pi.2^{50}$ a singularity of ction ingularity, before err	Fixup Return 1 Return max floating	radia <u>Unit</u> imum posi point num
	Input Arguments:  Type Scalar  Output Results: Type Scalar  Errors Detected: Error # 11 12  Comments: Error gets w Registers U	Precision DP  Precision DP  Argument outside of Argument too near the tangent function of the tangent function of the tangent as so that the tangent as so the tangent as the tangent as the tangent as the tangent as the tangent function of the tangent as the tan	How Pass F0  How Pass F0  Cause range   x   < π.2 <sup>50</sup> a singularity of ction ingularity, before err F0,F1,F2,F3,F4,F5.	Fixup Return 1 Return max floating or #12 is se	radia <u>Unit</u> imum posi  point num
	Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected:  Error # 11 12  Comments: Error gets v Registers U Algorithm: Multi	Precision  DP  Argument outside of Argument too near the tangent function rery large near a sinsafe Across Call:  Lely X by $\frac{4}{\pi}$ , and given the precision of the precision	How Pass  F0  How Pass  F0  Cause $\pi.2^{50}$ a singularity of ction ingularity, before err	Fixup Return I Return max floating or #12 is se	radia  Unita  imum posi point num nt.

singularity. The integer part of  $|x\cdot\frac{4}{\pi}|$  is the octant. If the octant is even, let w= fraction part of  $|x|\cdot\frac{4}{\pi}$  .

DTAN

Algorithm (Con't)

If the octant is odd, let w = -1 -fraction part of  $|x| \cdot \frac{4}{\pi}$ .

Next, compute two polynomials P(w) and Q(w). If  $w \ge 2^{-46}$ , then the

forms of the polynomials are:

$$P(w) = w(a_0 + a_1 w^2 + a_2 w^4 + w^6)$$

$$Q(w) = b_0 + b_1 w^2 + b_2 w^4 + b_3 w^6$$

If w <  $2^{-46}$ , then with u = w if  $|x| \cdot \frac{4}{\pi} < 1$ , and u = -w otherwise.

$$P(w) = w(a + u)$$

$$Q(w) = b_0 + b_3 u$$

where the values of the constants are:

 $a_0 = X'C58AFDD0A41992D4' = -569309.04006345$ 

 $a_1 = X'44AFFA6393159aa6' = 45050.3889630777$ 

 $a_2 = X'C325FD4A87357CAF' = -607.8306953515$ 

 $b_0 = x'C5B0F82C871A3B68' = -724866.7829840012$ 

 $b_1 = X'4532644B1E45A133' = 206404.6948906228$ 

 $b_2 = X'C41926DBBB1F469B' = -6438.8583240077$ 

 $b_3 = X'422376F171F72282' = 35.4646216610$ 

If  $w \le$  the comparand derived earlier and the octant = 1 or 2 (mod 4), then error 12 is sent. Otherwise, Q(w)/P(w) is returned with its sign adjusted. In octants = 0 or 3 (mod 4), P(w)/Q(w) is returned, with the sign adjusted according to tan(-x) = -tan(x).

The justification for this computation is that  $\frac{P(w)}{Q(w)} = \tan(w \cdot \frac{\pi}{4})$  and

$$\frac{Q(w)}{P(w)} = \cot(w \cdot \frac{\pi}{4})$$
, and simple trigonometric identities

give, for R = fraction part of X  $\cdot \frac{4}{\pi}$  :

DTAN

Algorithm (Con't)

Octant	(mod 4)	Formula for tan
0		$\tan( x ) = \tan(R \cdot \frac{\pi}{4})$
1		$\tan( x ) = \cot((1-R) \cdot \frac{\pi}{4})$
2	en e	$\tan( \mathbf{x} ) = -\cot(\mathbf{R} \cdot \frac{\pi}{4})$
3		$\tan( \mathbf{x} ) = -\tan((1-R) \cdot \frac{\pi}{4})$

which is the result of the computation as performed.

		DT ANH	
HZ	L/S-FC LIBRARY ROUTIN	NE DESCRIPTION	
Source Member Name:	DTANH Size	of Code Area: 94	Hw
Stack Requirement: _	22 Hw Data	CSECT Size:	0 Hw
☐ Intrins	ic 🛛	Procedure	
Other Library Module	s Referenced: DEX	<u> </u>	
ENTRY POINT DESCRIPT	TONS		
Primary Entry Name:	DTANH	المالية	
Function: Computes hyperbolic	tangent in double pr	الم يعنون في المناطقة	UCIBILITY OF L PAGE IS PO
Invoked by:			
TAN(X), where X	ed code for HAL/S con is a double precision	n scalar.	
Other Library	Modules:		
Execution Time (mic	coseconds): (See be	low)	
Input Arguments:			
<u>Type</u>	Precision	How Passed F0	Units
Scalar	DP.		
Output Results:			
<u>Type</u>	Precision	How Passed F0	Units
Scalar	DP.		
Errors Detected: Error #	Caus	i de la compania de La compania de la co	.xup
		현장 인계 등에 발생한 경기에 있다. - 1985년 1일	
Comments:			
Registers Unsaf	e Across Call: F0,F1	,F2,F3,F4,F5.	
		l or -1, according to the	
		DEXP), $tan( x ) = 1 -$	
Restore sign with	tan(-x) = -tanh(x).	For $ X  \leq 16^{-7}$ , $\tanh(x)$	) = X,
		(Cont'd. o	n next page)
Execution Time:  X  <	3.72529E-9: 47.8	하고 바다 하다 나는 사람들이 살아 있다. 	
3.72	3.72529E-9:47.8 529E-9 <  x  < 5.4931	E-1: 177.9	
	31E-1 <  X  < 2.0101E 01E + <  X  : 54.6	+1: 420.6	

DTANH

Algorithm (Con't)

For other values of x,  $16^{-7} < |x| < 0.54931$ , use a continued fraction

For other values of X, 16 
$$|X| < 0.54931$$
, use a conapproximation:  

$$\tanh(x) = x + x^3 \left( \frac{c_0}{x^2 + c_1 + c_2} - \frac{c_4}{x^2 + c_5} \right)$$

where the values of the constants are:

 $C_0 = X'COF6E12F40F5590A' = -.9643735440816707$ 

 $C_1 = X'419DA506FD3DBC84' = 9.8529882328255392$ 

 $C_2 = X'C31C504FEF537AF6' = 453.01951534852503$ 

 $C_3 = X'424D2FA31CAD8D0C' = 77.186082641955181$ 

 $C_4 = X'C3136E2A5891D8E9' = -310.8853383729134$ 

 $C_5 = X'4219B3ACA4C6E790' = 25.701853083191565$ 

		EA	TAN2	
HAL,	/S-FC LIBRARY RO	OUTINE DESCRIPTION		
Source Member Name:	EATAN2	Size of Code Area: 132		Hw
Stack Requirement:	18 Hw 1	Data CSECT Size:	10	_ Hw
☐ Intrinsic		Procedure		
Other Library Modules	Referenced:	None		
ENTRY POINT DESCRIPTION	ONS			
Primary Entry Name:	EATAN2			
Function: Computes a in single pr		actional approximation	in the rang	е (-т, т)
ARCTAN2(X,Y), wh	ere X and Y are	construct of the form single precision scal of the intended arcta	ars correspo	
Other Library Moo				
Execution Time (micro	seconds): 120	.0		
Input Arguments:  Type Scalar (sin) Scalar (cos)	Precision SP SP	How Passed FO F2		<u>Units</u>
Output Results:				
Type Scalar	Precision SP	How Passed FO		<u>Units</u> Radians
Errors Detected:				
Error # 62		Cause arg 2 = 0	Fixup Return 0.0	
Comments:				

Algorithm: The pointer to the data area that contains quadrant section constants is set and the sign of sin x is saved. The value  $Z = |\sin x|/|\cos x|$  is checked for several special cases. (1) If  $\cos x < 0$  and  $Z < (16)^{-6}$ , then return  $\pi \cdot \text{SIGNUM}(\text{SINX})$ . (2) If SINX = COSX = 0, then signal error and return 0. (3) If  $\text{SINX} \neq 0$ ,  $\text{COSX} = \emptyset$ , then return  $\pm \pi/2 \cdot \text{SIGNUM}(\text{Sin X})$ . (4) If  $\sin x \neq 0$ ,  $\cos x \neq 0$ , but  $Z > (16)^6$ , again return  $\pm \pi/2 = (\pi/2 \cdot \text{SIGNUM}(\sin x))$ . (5) If  $Z < (16)^{-3}$ , return Z.

Now, all of the special cases have been checked for. If the routine gets this far, it is time to reduce  $Z = \tan x$  so that  $Z < \tan \pi/12$  (tan 15°).

(Continued on next page)

EATAN2

Algorithm (Con't)

There are four cases to examine for Z in the 1st quadrant.

- A) Z > 1. Use 1/Z.
  - 1)  $\tan 15^{\circ} < 1/z \le 1$
  - 2)  $1/Z \leq \tan 15^{\circ}$
- B)  $Z \leq 1$ 
  - 3)  $tan 15^{\circ} < z \le 1$
  - 4) Z < tan 15°

For Z or  $1/Z > \tan 15^{\circ}$ , the reduction

$$\tan^{-1} z = \pi/6 + \tan(Y)$$
, where  $Y = Z \sqrt{3} - 1/Z + \sqrt{3}$  is used.

To protect significant bits, Y is computed as

$$Y = Z(\sqrt{3} - 1) - 1 + Z/Z + \sqrt{3}$$

Once Z or  $1/Z \le \tan 15^{\circ}$ , the formula for arctan Z can be applied.

$$\frac{\operatorname{Tan}^{-1}(Z)}{Z} = D + CZ^{2} + (B/(Z^{2} + A)), \text{ where the constants}$$

have the following values (hex values are used in the routine):

A = X'41168A5E' (1.4087812)

B = X'408F239C' (0.55913711)

C = X'BFD35F49' (-0.051604543)

D = X'409A6524' (0.60310579)

To adjust the angle to the proper section, the appropriate section constant is added to or subtracted from the intermediate result, as follows:

$$Z \le \tan 15^{\circ}$$
  $\rightarrow$  + 0 (E'0')  
 $\tan 15^{\circ} < Z \le 1$   $\rightarrow$  +  $\pi/6$  (X'40860A92')  
 $1/Z \le \tan 15^{\circ}$   $\rightarrow$  -  $\pi/2$  (X'Cll92lFB')  
 $\tan 15^{\circ} < 1/Z \le 1$   $\rightarrow$  -  $\pi/3$  (X'CLL0Cl52')

(Continued on next page)

ORIGINAL PLANT IS POOR

EATAN2

EATAN2

Algorithm (Con't)

We now have the correct angle for the first quadrant. All that remains is to fix the quadrant. If the  $\cos x < 0$ , then  $\tan^{-1}(x) = \pi - \tan^{-1}(z)$ . That fixes the angle to agree with the sign of  $\cos x$ . Now make the sign of the answer agree with the sign of  $\sin x$ , i.e.  $-\tan^{-1}(z)$  for  $-\sin x$  and  $+\tan^{-1}(z)$  for  $+\sin x$ .

The result, in radians, is in the correct quadrant in the range  $(-\pi, +\pi)$ .

EATAN2	
--------	--

Secondary E	ntry Name:	ATAN		
Function:	Computes arctan $(-\pi/2, +\pi/2)$ in		al approximation in n.	the range
Invoked by:				
X Compil	er emitted code for	or HAL/S constru	ct of the form:	
ARCTA	N(X), where X is a	single precisio	n scalar.	
Other	library modules:			
Execution I	ime (microseconds	): 116.5		
Input Argum	ments:			
<u>Type</u> Scalar	<u>Preci</u> S	<del></del>	How Passed F0	Units
Output Resu	Preci		How Passed	Units Radians
Scalar	in the second of the second	<b>P</b>	<b>FO</b>	Raulans
Errors Dete		Cause	<u><b>F</b>:</u>	ixup

### Comments:

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: Very similar to EATAN2, but the only special case that can be checked is  $Z = |\tan X| < (16)^{-3}$ . If Z is this small, then return Z to avoid an underflow exception later on. The algorithm for reduction and computation of  $Tan^{-1}$  Z is the same as EATAN2 again until quadrant fixing time. Since ARCTAN has only one arg, the result can only be adjusted in the range  $(-\pi/2, \pi/2)$ . The  $Tan^{-1}$  Z is computed for the first quadrant.

If the argument, tan x, is negative, the result is made negative.

		-	EPWRE	
HAL/S-	-FC LIBRARY ROUTIL	NE DESCRIPTION		
Source Member Name: E	PWRE Size	of Code Area:	32	_ Hw
		CSECT Size:	4	_ Hw
☐ Intrinsic	2	Procedure		
Other Library Modules Re	eferenced: EXP,	LOG, SQRT		· · · · · · · · · · · · · · · · · · ·
ENTRY POINT DESCRIPTIONS  Primary Entry Name: E	S EPWRE			
Function: Exponentiation of a sir	ngle precision sca	lar to a single p	recision scal	lar powe
Other Library Modu	econds): If Y =	.5: 124.7 .5: 337.1		
Input Arguments:	II 1 7			
	Precision SP SP	How Passe F0 F2	<u>a</u>	Unit -
Output Results:	Precision	How Passe	e <b>d</b>	Unit
<u>Type</u> Scalar	SP	FO		
Errors Detected:				
TITOTO DOCUMENT.	Cau		Fixup	
Error # 4 24	Zero raised Bas	to power < 0 e < 0	Return 0 use  bas	
24  Comments:  Registers Unsafe Ac	Base cross Call: F0,F1	e < 0 .,F2,F3,F4,F5.		
24  Comments:  Registers Unsafe Ac	Bas	e < 0 .,F2,F3,F4,F5.		

		EPWRI
HAL/S-FC LIBRARY ROL	JTINE DESCRIPTION	
Source Member Name: EPWRI Si	ze of Code Area: 38	Hw
Stack Requirement: 18 Hw Da	ita CSECT Size:	2 Hw
☐ Intrinsic	X Procedure	
Other Library Modules Referenced:	None	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: EPWRI		
Function: Exponentiation of a single precision integer power.	precision scalar to a	a double
Invoked by:  Compiler emitted code for HAL/S c	onstruct of the form	
X**I, where X is a single precisi	on scalar, and I is a precision int	
Other Library Modules:		
Execution Time (microseconds): (See	next page)	
Input Arguments:		
TypePrecisionScalar (base)SPInteger (exponent)DP	How Passed F0 R6	Units
Output Results:		i veri a per di interiori di probabili di distribi. Nga ang paggarang ang pag
Type Precision Scalar SP	How Passed F0	<u>Units</u>
Errors Detected:		
	use to power < 0	Fixup Return 0.0
Comments:		
Registers Unsafe Across Call: F0,	F1,F2,F3.	
Algorithm: Let I = exponent, E = base	. Write	
$I = \sum_{i=0}^{32} e_i 2^i, \text{ where e}$	i = 0 or 1 for all i.	
then $\mathbf{E}^{\mathbf{I}} = \mathbf{E}^{\mathbf{i}} \mathbf{e}_{\mathbf{i}} 2^{\mathbf{i}} \qquad \mathbf{e}_{\mathbf{i}} 2^{\mathbf{i}} \\ = \mathbf{\pi}_{\mathbf{i}} \mathbf{E}^{\mathbf{i}} = \mathbf{\pi}_{\mathbf{e}_{\mathbf{i}}} \mathbf{e}_{\mathbf{i}} \mathbf{e}_{$		and = 1 o+hoveise
4	전 : 12 12 12 13 13 13 13 13 13 13 13 13 13 13 13 13	-, Amm

### **EPWRI**

Execution Time:

38.2 + (n-1) 16.2 + 6.0m + 14.2 (if exponent negative),

where n = number of significant digits in binary representation of exponent.

Algorithm (Con't)

The product  $\pi_{e_i=1}^{2^i}$  is computed in a loop. Each time around the loop,  $E^{2^k}$  is multiplied by itself to give  $E^{2^{k+1}}$ . The k+l-st bit is shifted out of the exponent. If it is 1,  $E^{2^{k+1}}$  is multiplied into the result. If not, the result is left alone. When the remaining exponent is zero, the loop is finished and the result is  $E^I$ . If the exponent was positive, return  $E^I$ . Otherwise, return the reciprocal of  $E^I$ .

EPWRI	
DEMUT	

Secondary Entry Name: EPWRH		
Function: Exponentiation of a single precision scapower. Invoked by:	lar to a single precision	integer
X Compiler emitted code for HAL/S cons X**I, where X is a single precision integer	scalar, and I is a single	e precision
Other library modules:		
Execution Time (microseconds): Same as I	EPWRI, except constant is	38.8.
Input Arguments:  Type Precision  Scalar (base) SP  Integer (exponent) SP	How Passed FO R6	Units
Output Results:  Type Precision Scalar SP	How Passed F0	Units
Errors Detected:  Error # Caus  Caus  Zero raised to pow		
Comments: Registers Unsafe Across Call: F0,F1,	F2,F3.	
Algorithm: Halfword exponent is shift Then, EPWRI routine is use		fullword.

			EXP	
HAL.	/s-FC LIBRARY RO	UTINE DESCRIPTION	1	
Source Member Name: _	EXP S	ize of Code Area	: 108	Hw
Stack Requirement:		ata CSECT Size:	. 2	Hw
☐ Intrinsi		X Procedure		
Other Library Modules		None		
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name:	EXP single precision.	O.	EPRODUCIBILIT RIGINAL PAGE	
Invoked by:  X Compiler emitted  EXP(X), where X	l code for HAL/S		form:	
Other Library Mo TANH, EPWRE,  Execution Time (micro	SINH	8		
Input Arguments: <u>Type</u> Scalar	Precision SP	How P	assed	<u>Units</u>
Output Results:				is very silveri. Demokraji se s
<u>Type</u> Scalar	Precision SP	How F	assed	<u>Units</u>
Errors Detected:  Error #  6	and the second of the second o	Cause tside range:	Fixup Return maxin floating po	A 100 A
Comments:				
Registers Unsafe	Across Call: F	0,F1,F2,F3.		
and $0 \le T \le 1$ . Then	$\log_2 e = 4R-S-T$ , where $\log_2 e = 16^R \cdot 2^{-S}$ .	here R and S are $2^{-\mathrm{T}}$	integers, 0 ≤ S	< 3,
$2^{-T}$ is computed by a $2^{-T}=1$	fractional appr $ + \frac{2T}{CT^2 - T + D} $	oximation of the $\frac{B}{A + T^2}$	form:	

of the constants are:

The computation is carried out in fixed-point, and the values and scaling

EXP

Algorithm (Con't)

A = X'576AE119' = 87.417497 at bit 7

B = X'269F8E6B' = 617.97227 at bit 11

C = X'B9059003' = -0.03465736 at bit (-4)

D = X'B05CFCE3' = -9.95459578 at bit 4

The multiplication by  $2^{-S}$  is carried out by shifting right S places, and the multiplication of  $16^R$  is done by adding R to the floating exponent.

IPWRI	
HAL/S-FC LIBRARY ROUTINE DESCRIPTION	
at a Godo Aront 46	Hw
Source Member Name. 11 Water	2 Hw
Stack Requirement.	
Intrinsic Procedure	
Other Library Modules Referenced: None	
ENTRY POINT DESCRIPTIONS	
Primary Entry Name.	DUCEBILITY OF THE
Function:	
Computes double precision integer to positive double precipower.	
Invoked by:  X Compiler emitted code for HAL/S construct of the form:	
arg 1 ** arg 2, where arg 1 is a double precision integer arg 2 is a positive double precision	er, and ion literal.
Other Library Modules:	
k + 16.4(n-1) + 7.0m + 0.4(n-1)	-2) if n > 2, where
Execution Time (microseconds): k = 44.6, $n = #$ of significant digits in bir	nary representation of
arg2, m = # of significant ones in binary re Input Arguments:  Bregision How Passed	Units
Integer (base) DP R5	
Integer(exponent) DP R6	
Output Results: Type Precision How Passed	Units
Integer DP R5	
Errors Detected:  Error # Cause	Fixup
D1. L V	Return 0
Comments:  Registers Unsafe Across Call: R5.	
	t B = base,
Algorithm: Shift all halfwords to convert to fullwords. Le	
Write $I = \sum_{i=0}^{32} e_i 2^i$ , where $e_i = 1$ for each i. Then:	
$B^{I} = {\pi \atop B} B^{i}^{2} = {\pi \atop e_{i}} = 1^{B^{2}}$ if $e_{i} = 1$ , and $e_{i} = 1$ other	rwise.
$\mathbf{B}^{\dagger} = \mathbf{B}^{\dagger} \cdot \mathbf{B}^{\dagger} $	
그림은 어느 그는 사람은 그는 사는 사람들이 가는 사람들이 가는 사람들이 가는 사람들이 되는 것이다.	

(Continued on next page)

IPWRI

Algorithm (Con't)

The product  $\pi_{e_i}=1$   $B^{2^i}$  is computed in a loop. Each time around the loop,  $2^k$   $1^{k+1}$   $1^{k+1}$  is multiplied by itself to give  $1^k$   $1^k$   $1^k$   $1^k$   $1^k$   $1^k$   $1^k$   $1^k$   $1^k$  multiplied by  $1^k$   $1^k$ 

IPWRI

Secondary Entry Name: _	IPWRH		
Function: Computes double precis	sion integer t	o positive single precisi	on integer power.
Invoked by:			
Compiler emitted of arg 1 ** arg 2, wh  Other library modu	nere arg 1 is arg 2 is	construct of the form: a double precision intege a positive single precisi	r, and on integer literal.
Execution Time (microse	econds): Same	as for IPWRI, except k =	46.6.
Input Arguments:  Type  Integer(base)  Integer(exponent)	Precision DP SP	How Passed R5 R6	<u>Units</u>
Output Results: Type	Precision	How Passed	<u>Units</u>
Integer	DP	<b>R5</b>	
Errors Detected: Error #		Cause	<u>Fixup</u>
4	Zero r	raised to power <= 0	Return 0
Comments:  Registers Unsafe A	cross Call: I	R5.	
Algorithm:			
See IPWRI			

IPWRI	
-------	--

Secondary Entry Name:	HPWRH		
Function:			
Computes single p	recision integer to p	positive single precision inte	eger power
Invoked by:			
<pre>X Compiler emitted     arg 1 ** arg 2,</pre> Other library mod	arg 2 is a posi	ruct of the form: gle precision integer variable tive single precision position titeral.	e, and ve
Execution Time (micros	seconds): Same as for	r IPWRI, except k = 49.4.	
Input Arguments:  Type  Integer(base)  Integer(exponent)  Output Results:		How Passed R5 R6	Units
<u>Type</u> Integer	Precision SP	How Passed R5	Units -
Errors Detected: Error # 4	Cause Zero raised to power	Fixup <= 0. Return 0	
Comments: Registers Uns	afe Across Call: R5.		
Algorithm:			
See IPWRI.			

			L	og	
HAT	/s-FC LIBRARY	ROUTINE DESCRIPT	PION		
		Size of Code Ar		90	Hw
Source Member Name: _	LOG	Size of Code Ar	-		
Stack Requirement:	Hw	Data CSECT Size	<b>:</b>	2	Hw
☐ Intrinsi	. <b>c</b>	X Procedure			***
Other Library Modules	Referenced:	None			
ENTRY POINT DESCRIPTI	ONS				
Primary Entry Name: _ Function: computes log(X)		cision.	REPR ORIG	ODUCIBII NAL PAC	ITY OF THE SE IS POOR
Invoked by:  [X] Compiler emitted  LOG(X), where	l code for HAI X is a single	L/S construct of o	the form:		
X Other Library Mo	odules: ASII	NH, ATANH, EPWRE,	ACOSH		
Execution Time (micro	oseconds): 1	40.5			
Input Arguments:			Doggod		Units
<u>Type</u>	Precision	<u> 10</u>	w Passed		
			ΠO		
Scalar	SP		F0		
	SP		FO		
Scalar Output Results: Type Scalar	SP  Precision SP	<u>Ho</u>	FO  Passed FO		<u>Units</u>
Output Results: Type Scalar	Precision	Но	w Passed		Units
Output Results: Type	Precision	Ho Cause	w Passed	Fixup	
Output Results:  Type Scalar  Errors Detected:	Precision SP		w Passed FO	For X=0, for X <	Units  return LOG( 0, return negative floa
Output Results:  Type Scalar  Errors Detected: Error # 7  Comments:	Precision SP	<u>Cause</u> coutside range X	w Passed FO	<pre>For X=0, for X &lt; maximum point nu</pre>	return LOG( 0, return negative flo
Output Results:  Type Scalar  Errors Detected: Error # 7  Comments: Registers	Precision SP  argument s Unsafe Acros	Cause coutside range X ss Call: F0,F1,F2,	w Passed F0 > 0 F3,F4,F5	For X=0, for X < maximum point nu	return LOG( 0, return negative floa mber.
Output Results:  Type Scalar  Errors Detected: Error #  7  Comments: Registers  Algorithm: Write X 12 < M < 1. P, Q, a	Precision SP  argument  s Unsafe Acros $= 16^{P} \cdot 2^{-Q}$ nd M are found	Cause coutside range X ss Call: F0,F1,F2, M, where Pand Q d by fixed-point of	w Passed F0 > 0 F3,F4,F5 are inte	For X=0, for X < maximum point nu gers, 0 <	return LOG( 0, return negative floa mber.  Q < 3, and
Output Results:  Type Scalar  Errors Detected: Error #  7  Comments: Registers	Precision SP  argument  s Unsafe Acros $= 16^{P} \cdot 2^{-Q}$ nd M are found	Cause coutside range X ss Call: F0,F1,F2, M, where Pand Q d by fixed-point of	w Passed F0 > 0 F3,F4,F5 are inte	For X=0, for X < maximum point nu gers, 0 <	return LOG( 0, return negative floa mber.  Q < 3, and
Output Results:  Type Scalar  Errors Detected: Error #  7  Comments:  Registers  Algorithm: Write X $\frac{1}{2} \le M \le 1$ . P, Q, a  B = 0, if M > $\frac{\sqrt{2}}{2}$ ,  Let Z = $(M-A)/(M+A)$ Log( $(1+Z)/(1-Z)$ ) is	Precision SP  argument  S Unsafe Acros $= 16^{P} \cdot 2^{-Q}$ and A = $\frac{1}{2}$ , B  Then log(X computed by	Cause  coutside range X  as Call: F0,F1,F2,  M, where Pand Q  d by fixed-point of  = 1 otherwise.  ) = (4P-Q-B)log 2	w Passed F0 > 0 F3,F4,F5 are intecalculati + log((1	For X=0, for X < maximum point nu  gers, 0 < ons. Let  +2)/(1=2)	return LOG( 0, return negative flow mber. $Q \leq 3$ , and $A = 1$ ,
Output Results:  Type Scalar  Errors Detected: Error #  7  Comments:  Registers  Algorithm: Write X $\frac{1}{2} \le M \le 1$ . P, Q, a  B = 0, if M > $\frac{\sqrt{2}}{2}$ ,  Let Z = $(M-A)/(M+A)$ Log( $(1+Z)/(1-Z)$ ) is	Precision  SP  argument  SUnsafe Acros  = $16^P \cdot 2^{-Q}$ and M are found  and A = $\frac{1}{2}$ , B  Then log(X computed by $(\frac{Rw^2}{S-W^2})$	Cause  coutside range X  as Call: F0,F1,F2,  M, where Pand Q  d by fixed-point of  = 1 otherwise.  ) = (4P-Q-B)log 2  an approximation of	w Passed F0 > 0 F3,F4,F5 are intecalculati + log((1 of the fo	For X=0, for X < maximum point nu  gers, 0 < ons. Let  +2)/(1=2)	return LOG( 0, return negative flow mber. $Q \leq 3$ , and $A = 1$ ,

		s:	IN	<del></del>
H	AL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	SIN Siz	e of Code Area:	70	Hw
Stack Requirement:	0 Hw Dat	a CSECT Size:	30	Hw
X Intrin	sic	Procedure		
Other Library Mcdul	es Referenced: <u>None</u>			
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	SIN			
Function: Computes sine(X	() in single precision	n.		
	ed code for HAL/S co X is a single precis Modules:		n ;	
Execution Time (mic: Input Arguments: Type		<pre></pre>		
Input Arguments:	π < χ	or X <-π: 123.6		<u>Ur</u> radi:
Input Arguments: <u>Type</u> Scalar	X > π  Precision	or X <-π: 123.6 How Passed		
Input Arguments: Type	X > π  Precision	or X <-π: 123.6 How Passed		radi
Input Arguments:  Type Scalar  Output Results:  Type Scalar	Precision SP Precision	or X <-π: 123.6  How Passed F0  How Passed		
Type Scalar Output Results: Type	Precision SP Precision	or X <-π: 123.6  How Passed F0  How Passed F0		radi Ur
Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected: Error # 8  Comments: Called as Rl and R3, which ar	Precision SP  Precision SP  Cau argument outside	How Passed F0  How Passed F0  See of range:  er: uses only fixe com an intrinsic.	Fixup return V	radi
Input Arguments:  Type Scalar  Output Results:  Type Scalar  Errors Detected: Error # 8  Comments: Called as Rl and R3, which ar Registers	$\frac{\text{Precision}}{\text{SP}}$ $\frac{\text{Precision}}{\text{SP}}$ $\frac{\text{Cau}}{ \mathbf{x} } < \pi.2^{18}$ a library by compile a restored at exit for	How Passed F0  How Passed F0  How Passed F0  se of range:  er: uses only fixed com an intrinsic. 1,R3,R4,F0,F1,F2,F3	Fixup return // d-point re	radi

(Continued on next page)

SIN

Algorithm (Con't)

Since Q has been adjusted, it is only necessary to compute  $\sin(\frac{\pi}{4}(Q+R))$ .

If  $Q_0 = Q \mod 8$ , then this is equal to  $\sin(\frac{\pi}{4}(Q_0 + R))$ . The eight cases of this yield, through simple trigonometric identities:

$$Q_0 = 0: \sin(R \cdot \frac{\pi}{4})$$

1: 
$$\cos((1-R) \cdot (\frac{\pi}{4})$$

2: 
$$cos(R \cdot \frac{\pi}{4})$$

3: 
$$\sin((1-R) \cdot \frac{\pi}{4})$$

4: 
$$-\sin(R \cdot \frac{\pi}{4})$$

5: 
$$-\cos((1-R) \cdot \frac{\pi}{4})$$

6: 
$$-\cos(R \cdot \frac{\pi}{4})$$

7: 
$$-\sin((1-R) \cdot \frac{\pi}{4})$$

Let  $R_0 = R$  in octants 0, 2, 4, 6 and  $R_0 = 1-R$  in octants 1, 3, 5, 7.

We compute  $\sin(R_0 \cdot \frac{\pi}{4})$  in octants 0, 3, 4, 7 and  $\cos(R_0 \cdot \frac{\pi}{4})$  in octants 1, 2, 5, 6, and negate the result in octants 4, 5, 6, 7.

 $Sin(R_{o} \cdot \frac{\pi}{4})$  and  $cos(R_{o} \cdot \frac{\pi}{4})$  are computed by polynomial approximations.

SIN

( )

Algorithm (Con't)

The form of the approximation for sine is:

$$\sin(R_0 \cdot \frac{\pi}{4}) = R_0(a_0 + a_1R_0^2 + a_2R_0^4 + a_3R_0^6)$$

where the values of the constants are:

$$a_0 = X'40C90FDB' = .78539819$$

$$a_1 = X'C014ABBC' = .080745459$$

$$a_2 = X'3EA32F62' = .0024900069$$

$$a_3 = X'BD25B368' = -.000035943$$

The form of the approximatin for cosine is:

$$cos(R_0 \cdot \frac{\pi}{4}) = 1 + a_1 R_0^2 + a_2 R_0^4 + a_3 R_0^6$$

where the values of the constants are:

$$a_1 = X'C04EF4EE' = -.30842483$$

$$a_2 = X'3F40ED0F' = .0158510767$$

$$a_3 = X'BE14F17D' = -,000319570$$

SIN

Secondary Entry Name:	COS				
Function: Computes cosin	e(X) in single	precision.		·•	
Invoked by:					
X Compiler emitted	code for HAL	'S construct o	of the form:		
COS(X), where X	is a single p	recision scal	ar,		
Other library mo	dules:				
Execution Time (micro	seconds): -π X	$\leq x \leq \pi$ : 122 > $\pi$ or $x < -\pi$ :	.1 123.1		
Input Arguments:					
Туре	Precision		How Passed		Units
Scalar	SP		FO		radians
Output Results:					
<u>Type</u>	Precision		How Passed		Units
scalar	SP		FO		<del>-</del>
Errors Detected:					
Error #		Cause	18	Fixup	
8 + 11 - 11	argument outs	ide range  X	< π. 2	Return V2	•
			ı	2	
Comments:					

See SIN Comments.
Registers Unsafe Across Call: R1,R3,R4,F0,F1,F2,F3,F4,F5.

## Algorithm:

See SIN Algorithm.

				SINH	_
	HAL/	S-FC LIBRARY	ROUTINE DESCRIPTION	N	
in section of	Source Member Name:	SINH	Size of Code Area	.: 80	Hw
	Stack Requirement:	18 Hw	Data CSECT Size:	2	Hw
	☐ Intrinsic		☑ Procedure		
	Other Library Modules	Referenced:	EXP		
<del> </del>	ENTRY POINT DESCRIPTION	ONS			
	Primary Entry Name: Function: Computes hyperbolic	SINH sine in singl		REPRODUCIBILI' ORIGINAL PAGI	TY OF THE
	Invoked by:  X Compiler emitted  SINH(X), where X		/S construct of the precision scalar.	e form:	
	Other Library Mo	dules:			
	Execution Time (micro	seconds): 2	$ x  \le  x  \le 1.75366E + 2.0394E - 4 \le  x  \le 1.866E + 2.0394E - 4 \le 40.0394E - 4 $	80.7	
	Input Arguments:  Type  Scalar	Precision SP	How F	Passed O	Units
	Output Results: <u>Type</u> Scalar	Precision SP	How F	Passed 0	Units
	Errors Detected:			Fixup	
	<u>Error #</u> 9		<u>Cause</u> ent outside range 175.366	Return maxim floating poi	um positive
	Comments:				
	Registers Unsafe		F0,F1,F2,F3,F4,F5		
	Algorithm: If X < 2. then sinh(x) is compu	04E-4, then sited via a po	sinh(x) = x. If 2. lynomial <b>a</b> pproximat	$04E-4 \le  X  < 1$ ion.	
	The form of the poly	nomial is:			
	$\sinh(x) = X + C_1^2$	$x^3 + c_3 x^5 + c_3$	<b>3<sup>7</sup></b>		
	where the values of	the constants	are:		

SINH

Algorithm (Con't)

$$C_1 = X'402AAAB8' = 0.16666734$$

$$C_2 = X'3F221E8C' = 0.008329912$$

$$C_3 = X'3DD5D8B3' = .2039399E-3$$

Otherwise, sinh(|x|) or cosh(|x|) is calculated using EXP. The number V, equal to 0.4995050, is introduced to control rounding errors and the formula is as follows:

$$\sinh(x) = \frac{1}{2V} \left( e^{(x+\log v)} - \frac{v^2}{e^{(x+\log v)}} \right)$$

$$\cosh(x) = \frac{1}{2V} (e^{(x+\log v)} + \frac{v^2}{e^{(x+\log v)}})$$

the identities sinh(-x) = -sinh(x) and cosh(-x) = cosh(x) are used to recover sinh(x) and cosh(x) from sinh(|x|) and cosh(|x|).

SIL	TI-I	
 $\sim$ $^{-1}$	ATT	

Secondary Entry Name	e: COSH		
Function:	ic cosine in single precision		
	re cosine in single precision		
Invoked by:			
	ed code for HAL/S construct K is a single precision scal		
Other library	modules:		
Execution Time (mic	coseconds): 228.9		
Input Arguments:  Type  Scalar	Precision SP	How Passed F0	Units
Output Results:			
Type	Precision	How Passed	Units
Scalar	<b>SP</b>	FO	
Errors Detected:			
Error #	<u>Cause</u>	Fix	qu
	Argument outside range $ x  \le 175.366$		ximum positive point number.
Comments:			
Registers Unsaf	e Across Call: F0,F1,F2,F3,F	<b>∕</b> 1 ⋤5	
		<b>르셔딩</b> 하다 그 모양 등을 되었	

## Algorithm:

See SINH algorithm.

•	Approximately and the second s			
		S	ORT	
HARACTER STATE OF THE BOOK OF	AL/S-FC LIBRARY ROUTINE	DESCRIPTION		
Source Member Name:	SQRT Size o	f Code Area:	48	Hw
Stack Requirement: _	0 Hw Data C	SECT Size:	14	Hw
X Intrins	sic 🔲	Procedure		
Other Library Module	es Referenced: None			
ENTRY POINT DESCRIPT	FIONS			
Primary Entry Name: Function:	SQRT	REPROL	UCIBIL 1777	OF THE
	root in single precisio	n.		
Invoked by:				
	ed code for HAL/S const e X is a single precisi			
v Other Library !	Modules: ACOS, ASINH,	EPWRE. VV10S3	ACOSH	
Execution Time (mic	roseconds): 88.3			
Input Arguments:				
Tilde in Amiene				
Type Scalar	Precision SP	How Passed FO		<u>Units</u>
<u>Type</u> Scalar	<del></del>	· · · · · · · · · · · · · · · · · · ·		Units
Type	<del></del>			
<u>Type</u> Scalar Output Results: <u>Type</u>	SP Precision	F0 How Passed		
<u>Type</u> Scalar Output Results: <u>Type</u> Scalar	SP Precision	F0 How Passed	i Fixup	Units
Type Scalar  Output Results: Type Scalar  Errors Detected:	Precision SP	How Passed		Units Units
Type Scalar  Output Results: Type Scalar  Errors Detected: Error # 5  Comments:	Precision SP  Cause Argument outs X > 0	How Passed FO	Fixup Return SQR	Units
Type Scalar  Output Results: Type Scalar  Errors Detected: Error # 5  Comments:	Precision SP  Cause Argument outs	How Passed FO	Fixup Return SQR	Units
Type Scalar  Output Results: Type Scalar  Errors Detected: Error # 5  Comments: Registers Un	Precision SP  Cause Argument outs X > 0	How Passed F0 side range	Fixup Return SQR 0, F1, F2,	<u>Units</u>

(Continued on next page)

 $\sqrt{M}$  by a hyperbolic approximation.

SQRT

Algorithm (Con't)

The form of the approximation is, for Q=0

$$\sqrt{M} = a - \frac{b}{(\frac{C}{2}) + (\frac{M}{2})}$$
  $\left[\frac{C}{2} + \frac{M}{2} \text{ is to avoid fixed-point overflow for large M}\right]$ 

where the calculations are done in fixed-point.

The values of the constants are:

a = X'OlAE7D00' = 1.6815948 at bit 7

b = X'FF5B02F1' = -1.2889728 at bit 7

 $\frac{C}{2}$  = X'35CFC610' = 0.42040325 at bit 0

For Q = 1,  $\frac{a}{4}$  and  $\frac{b}{4}$  are used instead of a and b.

 $\frac{a}{4} = X'006B9F40' = 0.4203987$  at bit 7

 $\frac{b}{4} = X'FFD6C0BD' = -0.3222432$  at bit 7

The first approximation is improved with two passes of the Newton-Raphson iteration. The form of the first is:

$$y_1 = \frac{1}{2}(\frac{x}{y_0} + y_0)$$

The form of the second, to minimize truncation errors, is:

$$y_2 = \frac{1}{2}(\frac{x}{y_0} - y_0) + y_0$$

y, is returned as the answer.

			TAN	<b>-</b>
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
			310	<b>**</b>
Source Member Nam	ne: TAN	Size of Code Area:	112	Hw Hw
Stack Requirement	t: <u>20</u> Hw	Data CSECT Size:	4	Hw
☐ Int:	rinsic	Procedure		
Other Library Mod	dules Referenced	None		
ENTRY POINT DESCR	RIFUIONS			
The state of the s	<b></b>			
Primary Entry Nar	me: TAN			
Function. Computes	tangent in single	precision.		
	en e			
Invoked by:				
		's construct of the f	orm;	
TAN(X), whe	ere X is a single p	recision scalar.		
				garante de la companya de la company
Other Libra	ry Modules:			
Other Libra	ry Modules:			
Other Libra		64.0		
Execution Time (	microseconds): 10	54.0		
Execution Time (	microseconds): 16	64.0 How Pas	sed	<u>Unit</u>
Execution Time () Input Arguments: Type	microseconds): 16		<del></del>	
Execution Time (	microseconds): 16	How Pas	<del></del>	· · · · · · · · · · · · · · · · · · ·
Execution Time () Input Arguments: Type Scalar	microseconds): 16	How Pas	<del></del>	
Execution Time () Input Arguments:	microseconds): 16  Precision SP	How Pas		radia
Execution Time ()  Input Arguments:  Type  Scalar  Output kesults:  Type	Precision  Precision	How Pas	ssed	radia
Execution Time () Input Arguments:	microseconds): 16  Precision SP	How Pas	ssed	radia
Execution Time ()  Input Arguments:  Type  Scalar  Output kesults:  Type	Precision SP  Precision SP	How Pas  How Pas  F(	ssed	radia
Input Arguments:  Type Scalar  Output kesults:  Type Scalar	Precision SP  Precision SP  Precision SP	How Pas  F(  How Pas  F(	ssed ) Fixup	radia
Input Arguments:  Type Scalar  Output kesults:  Type Scalar  Errors Detected: Error #	Precision SP  Precision SP  Argument outside	How Pas  F(  How Pas  F(  Cause range   X   < π.2 <sup>18</sup>	Ssed Fixup Return 1	radia Unit
Input Arguments:  Type Scalar  Output kesults:  Type Scalar  Errors Detected: Error #	Precision SP  Precision SP  Argument outside Argument too clos	How Pase  How Pase  F(  Cause $ x  < \pi.2^{18}$ e to singularity of	Fixup Return l Return ma	radia Unit
Input Arguments:  Type Scalar  Output kesults:  Type Scalar  Errors Detected:  Error #  11 12	Precision SP  Precision SP  Argument outside	How Pase  How Pase  F(  Cause $ x  < \pi.2^{18}$ e to singularity of	Fixup Return l Return ma	radia Unit
Input Arguments:  Type Scalar  Output kesults:  Type Scalar  Errors Detected:  Error #  11 12  Comments:	Precision SP  Precision SP  Argument outside Argument too clos tangent functio	How Pase  How Pase  F(  Cause $ x  < \pi.2^{18}$ e to singularity of	Fixup Return 1 Return ma	radia Unit
Input Arguments:  Type Scalar  Output kesults: Type Scalar  Errors Detected: Error #  11 12  Comments: Register	Precision SP  Precision SP  Argument outside Argument too clos tangent function rs Unsafe Across Ca	How Pas  F(  How Pas  F(  A How Pas	Fixup Return 1 Return ma	Unitation of the control of the cont
Input Arguments:  Type Scalar  Output kesults: Type Scalar  Errors Detected: Error #  11 12  Comments: Register	Precision SP  Precision SP  Argument outside Argument too clos tangent function rs Unsafe Across Ca	How Pas  F(  How Pas  F(  Cause  range   x   < π.2 <sup>18</sup> e to singularity of  n	Fixup Return 1 Return ma	radia Unit  wimum pos point nu

(Continued on next page)

5-132

TAN

Algorithm (Con't)

We have the following table:

Q mod 4	tan(x)
0	$\tan( \mathbf{x} ) = \tan(\mathbf{R} \cdot \frac{\pi}{4})$
	$\tan( \mathbf{x} ) = \cot((1-R) \cdot \frac{\pi}{4})$
2	$\tan( \mathbf{x} ) = -\cot(\mathbf{R} \cdot \frac{\pi}{4})$
	$\tan( x ) = -\tan((1-R) \cdot \frac{\pi}{4})$

For  $Q \mod 4$  even, let w = R, and for  $Q \mod 4$  odd, let w = 1-R. Compute two polynomials in w, as polynomials in

$$u = \frac{w^2}{2} :$$

$$P(w) = w(a_0 + u)$$

$$Q(w) = b_0 + b_1 u + b_2 u^2$$

then  $tan(w) = \frac{P(w)}{Q(w)}$ ,  $cot(w) = \frac{Q(w)}{P(w)}$ , and the above table describes how

tan(x) is computed. Finally, tan(x) is computed using the identity tan(-x) = -tan(x).

The values of the constants are:

$$a_0 = X'C1875FDC' = -8.4609032$$

$$b_0 = X'ClAC5D33' = -10.7727537$$

$$b_1 = X'415B40FD' = 5.7033663$$

$$b_2 = x' c028c93F' = -.15932077$$

		-	TANH	
на	L/S-FC LIBRARY H	ROUTINE DESCRIPTION		
Source Member Name:	TANH	Size of Code Area:	56	Hw
Stack Requirement: _	18 Hw	Data CSECT Size:	0	Hw
☐ Intrins	ic	x Procedure		
Other Library Module	s Referenced:	EXP	· · · · · · · · · · · · · · · · · · ·	
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	TANH			
Function: Computes hyperbolic	tangent in sing	le precision.	REPRODUC ORIGINAL	CIBILITY OF TH PAGE IS POOR
		S construct of the precision scalar.	form;	
Other Library M	odules:			
	2			
Execution Time (micr 2.4414E-4	$<  X  \le 7.0E-1$ :	78.7, 7.0E-1 <  x	< 9.011: 224	. <b>. 4</b>
Input Arguments:	9.	$011 \le  X  : 42.4$		en en de la deservación de la company. O de la company de la comp
<u>Type</u> Scalar	Precision SP	How Pas	ssed	Units -
Output Results: Type	Precision	How Pas	sseđ	Units
Scalar	SP	<u> </u>	<u> </u>	<u> </u>
Errors Detected:				
Error #		Cause	Fixup	
Comments: Registers Unsaf	e Across Call: F	0,F1,F2,F3,F4,F5.		
Algorithm: If $  x   \le   x   \le 9.011$	x  > 9.011, retu	(x) + 1 or $-1$ , accorded $(x) + 1$ or $-1$ , accorded $(x) + 1$	ing to the s $/(1+e^2 x )$ .	ign of X. Restore
sign with $tanh(-x) =$	-tanh(x). For	$ x  \leq 16^{-3}, \tanh(x)$	= X.	
For other values of	$x, 16^{-3} <  x  < 0$	.7, use a rational	approximatio	n.
where the values of				
a = X'BEF7EA70' =	003782895			
b = X'COD08756' =	81456511			

# 5.3.3 Vector/Matrix Routine Descriptions

This subsection presents a class of routines which deal with HAL/S vector/matrix operations. Some of the routines implement HAL/S language built-in functions while others implement HAL/S operators.

			MMODNP	<u> </u>
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Na	me: MMODNP	Size of Code Area:	12	Hw
Stack Requiremen	t: 0 Hw	Data CSECT Size: _	0	Hw
💢 Int	rinsic	<pre>Procedure</pre>		
Other Library Mo	dules Referenced:	none.		
ENTRY POINT DESC	RIPTIONS			
Primary Entry Na	me: MMODNP			
	precision scalar to precision matrix.	o all positions in an	n x m partit	ion of a
Invoked by: Compiler em	nitted code for HAL	/S construct of the s a double precision	form: matrix.	
Other Libra  Execution Time (	mry Modules:	.8 + n(4.0 + 8.0m)		
Input Arguments:				
Type Scalar	Precision DP	How Pa	ssed	Uni
Integer (n) Integer (m) Integer (out	SP SP del) SP	R5 R6 R7		
Output Results: Type	Precision	How Pa	ıssed	Uni
Matrix (n,m	) DP	$R1 \rightarrow 0$	th element	•
Errors Detected:				
		<u>Cause</u>	Fixup	
Error #				
	store Uncafo Acros	a Calle pl po p4	DF D6 D7 T	ıń mi
	sters Unsafe Across	s Call: Rl, R3, R4,	R5, R6, R7, F	O, Fl.

			MMOSNP	
HAL/S	-FC LIBRARY F	ROUTINE DESCRIPT	TION	
Source Member Name: M				Hw Hw
Stack Requirement:				Hw Hw
X Intrinsic		Procedure		
Other Library Modules R	eferenced: _	None.		
ENTRY POINT DESCRIPTION				
Primary Entry Name:	MMOSNP			
Function: Fill an n x m partiti precisi	on of a singl	le precision mat	rix with a sin	gle
Invoked by:  [X] Compiler emitted of  *  MA TO B,C TO D = 0	code for HAL/ where M i	'S construct of s a single preci	the form:	
Other Library Mod		+ n(4.4 + 6.4m)		
Input Arguments:	Precision	<b>H</b> o	ow Passed	Units
Type Scalar Integer(n) Intger(m) Integer(outdel)	SP SP SP SP		F0 R5 R6 R7	
Output Results: Type Matrix (n,m)	Precision SP		ow Passed → 0 <sup>th</sup> element	<u>Units</u>
Errors Detected: Error #		<u>Cause</u>	<u>Fi</u>	xup
Comments: Registers	Unsafe Across	s Call: R1, R3,	R4, R5, R6, R	7, FO, F1.
In ma At exi is inc	ner loop sele trix and move	ps, one on n; onects row and column and colu	into location, matrix element	nd

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

		·
	MMlDNP	
HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
Source Member Name: MM1DNP	Size of Code Area: 18	Hw
Stack Requirement: 0 Hw	Data CSECT Size: 0	Hw
☐ Intrinsic	☐ Procedure	
 Other Library Modules Referenced:	None.	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: MMlDNP		
Function: Moves a partition of a	matrix in double precision.	
Invoked by:  X Compiler emitted code for HAL,  M2=Ml A TO B,C TO D; where M  A TO B,C TO D=Ml;  Other Library Modules:	/S construct of the form:  l and M2 are double precision matrice	es.
Input Arguments:  Type  Precision	8 + n(5.4 + 12.2m) for n x m matrix 1	
Matrix (n,m) DP Integer (n) SP Integer (m) SP Integer (indel,outdel) DP Output Results:	How Passed R2 → Oth element R5 R6 R7(indel in highest HW, outdel in Low HW)	Units
Type Precision	How Passed	Units
Matrix(n,m) DP	Rl → 0 <sup>th</sup> element	
Errors Detected:		
Comments: Registers Unsafe Across Co	Cause Fixup  all: R1, R2, R3, R4, R5, R6, R7, F0 F2, F3.	, Fl,
		out point

HAL/S-FC LIBRARY ROUTINE DESCRIPTION  Source Member Name: MMISNP Size of Code Area:  Stack Requirement: 0 Hw Data CSECT Size:  [X] Intrinsic Procedure  Other Library Modules Referenced: None,  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MMISNP  Function: Moves a partition of a single precision matrix  Invoked by:  [X] Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D' where M1 and M2 are single M2 A TO B, C TO D matrices;  [M2] Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  Type Precision R5  Matrix(n,m) SP R5	O	HW
Source Member Name: MMISNP Size of Code Area:  Stack Requirement: 0 Hw Data CSECT Size:  [X] Intrinsic	O	
Stack Requirement: 0 Hw Data CSECT Size:	O	HW
Other Library Modules Referenced:  None,  ENTRY POINT DESCRIPTIONS  Primary Entry Name:  MMISNP  Function: Moves a partition of a single precision matrix  Invoked by:  X Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D where M1 and M2 are single M2 A TO B, C TO D matrices.  Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  Type  Matrix(n m)  Precision  Precision  How Pass R2 + Oth		
Other Library Modules Referenced:  None,  ENTRY POINT DESCRIPTIONS  Primary Entry Name:  MMISNP  Function: Moves a partition of a single precision matrix  Invoked by:  X Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D where M1 and M2 are single M2 A TO B, C TO D matrices.  Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  Type  Matrix(n m)  Precision  Precision  How Pass R2 + Oth	<b>:</b>	
ENTRY POINT DESCRIPTIONS  Primary Entry Name: MMISNP  Function: Moves a partition of a single precision matrix  Invoked by:  [X] Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D' where M1 and M2 are single M2 A TO B, C TO D matrices.  [M2] Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  [M3] Type	<b>:</b>	
Primary Entry Name: MMISNP  Function: Moves a partition of a single precision matrix  Invoked by:  X Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D' where M1 and M2 are single M2 A TO B, C TO D matrices:  Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  Type Precision How Pass R2 + Oth Matrix(n, m)	<b>:</b>	
Function: Moves a partition of a single precision matrix  Invoked by:    X Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D' where M1 and M2 are single M2 A TO B, C TO D matrices.    Other Library Modules:    Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:   Type	<b>:</b>	
Compiler emitted code for HAL/S construct of the form M2 = M1 A TO B, G TO D' where M1 and M2 are single M2 A TO B, C TO D matrices.  Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for Input Arguments:  Type  Matrix(n,m)  Precision  R2 + Oth  Matrix(n,m)	m: precision	
Other Library Modules:  Execution Time (microseconds): 10.8 + n(5.4 + 9.4m) for  Input Arguments:  Type Precision R2 + 0th R2 + 0th		
Input Arguments:  Type Precision How Pass R2 + 0th	n z m matr	ix.
Integer(n) Integer(m) Integer(m) Integer(indel, outdel) Output Results:  Type  Precision  R6 R7 (high HW=ir How Pass	ed element del,Low HW= sed	Units
Matrix(n,m) SP $RI \rightarrow 0^{-1}$ e.	ement	
Errors Detected: Error # Cause	Fixup	
Comments: Registers Unsafe Across Call: R1,R2,R3,R4,R5	R6,R7,F0,F	
Algorithm: Loop on # rows;  Loop on # columns;  Load and store current element point increment pointers to next row;  End column loop;  Increment input pointer by indel;	to by input	t/output poin

-	MMlTNP
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION
	Source Member Name: MM1TNP Size of Code Area: 16 Hw
	Stack Requirement: 0 Hw Data CSECT Size: 0 Hw
	Intrinsic Procedure
	Other Library Modules Referenced: None.
	ENTRY POINT DESCRIPTIONS
	Primary Entry Name: MMlTNP
	Function:  Moves a partition of a double precision matrix and stores it as single precision.
	Invoked by:  [X] Compiler emitted code for HAL/S construct of the form:  M2=M1  A TO B, C TO D; where M2 is a single precision matrix.
	M <sup>2</sup> A TO B, C TO D' where Ml is a double precision matrix.  Other Library Modules:
	Execution Time (microseconds): 10.4 + n(5.8 + 10.6m) for n x m move.
	Input Arguments:  Type  Matrix(n,m)  DP  R2 $\rightarrow$ 0th element  R5
	Matrix(n,m) DP R2 → Oth element Integer(n) SP R5 Integer(m) SP R6 Integer (indel,outdel) DP R7  Output Results:
	Errors Detected:  Error # Cause Fixup
	Comments: Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6,R7,F0,F1,F2,F3.
	Algorithm: Loops on # rows;
	Loops on # columns; Load long input element pointed to by input pointer; Store short into output element pointed to by output pointer; Increment pointer to next element; End column loops; Increment input pointer by indel; Increment output pointer by outdel;
	End row loop;

				MM	LWNB	
	HAL/S	5-FC LIBRARY	ROUTINE DESC	RIPTION		
Source Membe	er Name:	MMLWNP	Size of Cod	e Area:	18	_ Hw
Stack Requir	rement:	O Hw	Data CSECT	Size:	0	_ Hw
G	Intrinsic		☐ Proc	edure		
Other Libra	ry Modules F	Referenced:	None			
ENTRY POINT	DESCRIPTION	NS.				
Primary Enti	ry Name:	MM1WNP				
	loves a part		ingle precisi	on matrix and	i stores it	as
			/S construct			
A TO	TO B, C TO  B,C TO D  Library Modu	, ,	Ml is a singl M2 is a doubl	e precision : e precision :	matrix; matrix.	
Execution T	ime (microse	econds): 13	.6 + n(5.0 +	ll.Om) for n	x m move.	
Input Argume	ents:	Dwo-i		Tavi Dalama		•••
Type Matrix ( Integer Integer Integer	(n)	Precision SP SP SP el) DP	R2	How Passed  Oth elemen  R5  R6  R7	nt	Uni
Output Resul						
<u>Type</u> Matrix(	(n,m)	Precision DP	RL	How Passed → Oth elemen		<u>Uni</u>
Errors Detec			Cause		Fixup	
DITOL					<u></u>	
Comments: R	legisters Un:	safe Across (	Call: Rl,R2,	R3,R4,R5,R6,	R7,F0,F1,F2	,F3.
	Loops on # 1 Loops on Load s	rows; # columns; short input e long (with 2	cating point element point zeroed second	ed to by inpu	ıt pointer;	ent
	End colum	mn loop;	s to next row	element;		
		t output poir	nter by outde	1,		

	мме	6DN	
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION		
	Source Member Name: MM6DN Size of Code Area:	42	Hw
	Stack Requirement: 0 Hw Data CSECT Size:	0	- Hw
	X Intrinsic Procedure		
	Other Library Modules Referenced: None.		
<del></del>	ENTRY POINT DESCRIPTIONS		
	Primary Entry Name: MM6DN	ing the second of the second o	
	Function: Multiplies two double precision matrices.		
	Invoked by:  Compiler emitted code for HAL/S construct of the form:		
	M1 M2, where M1 is a m x n double precision matrix	$m,n,l \neq 3$	
	M2 is a n x ℓ double precision matrix ∫  Other Library Modules:		
	Execution Time (microseconds): $22.2 + m(10.8 + \ell(21.2 + 27.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2.4 + 2$	2n))	
	Input Arguments: Type Precision How Passed		Units
	matrix (m,n) DP R2 → Oth element		-
	matrix $(n, \ell)$ DP R3 $\rightarrow$ 0 <sup>th</sup> element (Continued on next page)	<b>E</b>	
	Output Results:		
	Type Precision How Passed  Matrix $(m, \ell)$ DP R1 $\rightarrow$ 0 <sup>th</sup> element		Units
	RI O element		
	Errors Detected:		
	Error #	Fixup	
	성진 아들 회사회 사람이 모양하는 인상이 되는 하는 하는 사람들이 이렇게		
	Comments: Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6,F	R7,F0,F1,F2	,F3,F4,F
	Algorithm: Uses 3 loops:  Innermost (on n) multiplies a row of M1 by a  The second loop (on $\ell$ ) resets the column poir  The outer loop (on m) resets the row pointer.	nter;	M2;

### MM6DN

### MM6DN

Input Arguments (Con't):

Type		Precision	n	How Pa	ssed	Units
			<del>-</del>			- <del> </del>
Integer	(m)	SP		R5		
		SP		R6		
Integer	(l)	SP		R7		-

Source Member Name: MM6D3	Size of Code Area: 32	Hw
	Data CSECT Size: 0.	Hw
XX Intrinsic	Procedure	
Other Library Modules Referenced:	None.	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: MM6D3		
Function: Multiplies two 3 x 3 dou	ble precision matrices,	
runction. Multiplies two 3 x 3 do.		
Invoked by:  Compiler emitted code for HAL/S	construct of the form:	•
X Compiler emitted code for may	while precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do  Other Library Modules:	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do	puble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671	ouble precision 3 x 3 matrices.	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:	.6  How Passed	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision	ouble precision 3 x 3 matrices.  .6  How Passed $R2 \rightarrow 0^{th}$ element	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision	.6  How Passed	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision  Matrix (M1) DP  Matrix (M2) DP	ouble precision 3 x 3 matrices.  .6  How Passed $R2 \rightarrow 0^{th}$ element	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision  Matrix (M1) DP  Matrix (M2) DP  Output Results:	ouble precision 3 x 3 matrices.  .6 $\frac{\text{How Passed}}{\text{R2} \rightarrow \text{0th element}}$ R3 $\rightarrow$ 0th element	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision  Matrix (M1) DP  Matrix (M2) DP  Output Results:  Type Precision	ouble precision 3 x 3 matrices.  .6  How Passed $R2 \rightarrow 0^{th}$ element	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision  Matrix (M1) DP  Matrix (M2) DP  Output Results:	ouble precision 3 x 3 matrices.  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed	
M1 M2, where M1 and M2 are do  Other Library Modules:  Execution Time (microseconds): 671  Input Arguments:  Type Precision  Matrix (M1) DP  Matrix (M2) DP  Output Results:  Type Precision	ouble precision 3 x 3 matrices.  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed	

Algorithm: Explicitly nultiplies row by column, element by element.
Uses BCTB to advance to each new col., and BCTB to advance to each new row.

				MM6SN	
				- ON	
	HAL,	/S-FC LIBRARY	ROUTINE DESCRIPT	LON	
Source	Member Name:	MM6SN	Size of Code Are	ea: 40	Hw Hw
Stack R	equirement:	0 Hw	Data CSECT Size	0	Hw
	🔀 Intrinsi		☐ Procedure		
Other I	ibrary Modules		None.		
ENTRY F	OINT DESCRIPTI	ons			
Primary	Entry Name: _	MM6SN			
Function			recision matrices		
1 4	Multiplies	two single p	recratou macrices		
eriente en la T <b>it</b> ta e			/S construct of t		<b>~</b> 2
o <sub>1</sub>	M1 M2. where M	ll is a m x n 12 is a n x l	single precision single precision s	matrix, \ m n p	<b>≠ 3.</b>
	M1 M2, where M M ther Library Mo	il is a m x n 12 is a n x l dules:	single precision s single precision s	matrix, m,n,l	<b>≠ 3.</b>
	M1 M2, where M M ther Library Mo	il is a m x n 12 is a n x l dules:	single precision	matrix, m,n,l	<b>≠ 3.</b>
Execut	M1 M2, where M  ther Library Mo  ion Time (micro	il is a m x n il is a n x l odules: oseconds): 2	single precision principle single precision pr	matrix, m,n, l m,n, l matrix 20.2 + 18.0n)),	
Execut	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:	il is a m x n 12 is a n x l dules:	single precision is single precision in singl	matrix, m,n,l matrix m,n,l  20.2 + 18.0n)),  Passed Oth element	
Execut Input	M1 M2, where M  ther Library Mo  ion Time (micro	ll is a m x n  12 is a n x l  adules:  aseconds): 2	single precision is single precision in singl	matrix, m,n, l matrix m,n, l 20.2 + 18.0n)),	
Execut Input Ma Ma	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Fype  trix (m,n)  trix (n,l)  (Continue	ll is a m x n  12 is a n x l  adules:  bseconds): 2  Precision  SP	single precision is single precision in singl	matrix, m,n,l matrix m,n,l  20.2 + 18.0n)),  Passed Oth element	
Execut Input Ma Ma Output	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Type  trix (m,n)  trix (n,l)  (Continue  Results:	il is a m x n il is a m x n il is a n x l odules:  oseconds):  Precision SP SP ed on next page	single precision is single precision in singl	matrix, m,n,l matrix m,n,l  20.2 + 18.0n)),  Passed Oth element	Unit
Execut Input Ma Ma Output	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Fype  trix (m,n)  trix (n,l)  (Continue	il is a m x n il is a m x n il is a n x l odules:  pseconds): 2  Precision SP SP	single precision is single precision in singl	matrix, m,n,l  20.2 + 18.0n)),  Passed Oth element Oth element	Unit
Execut Input Ma Ma Output Ma	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Type  trix (m,n)  trix (n,l)  (Continue  Results:	lisamxn lisamx	single precision is single precision in singl	matrix, m,n,l matrix m,n,l 20.2 + 18.0n)), Passed Oth element Oth element Oth element	Unit
Execut Input Ma Ma Output Ma Errors	M1 M2, where M M ther Library Mo ion Time (micro Arguments: Type trix (m,n) trix (n,l) (Continue Results: Type trix (m,l)	lisamxn lisamx	single precision is single precision in singl	matrix, m,n,l matrix m,n,l  20.2 + 18.0n)),  Passed Oth element Oth element	Unit
Execut Input Ma Ma Output Ma Errors	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Fype  trix (m,n)  trix (n,l)  (Continue  Results:  Type  trix (m,l)  Detected:	lisamxn lisamx	single precision is single precision in singl	matrix, m,n,l matrix m,n,l 20.2 + 18.0n)), Passed Oth element Oth element Oth element	Unit
Execut: Input  Ma  Ma  Output  Ma  Errors	M1 M2, where M  ther Library Mo  ion Time (micro  Arguments:  Type  trix (m,n)  trix (n,l)  (Continue  Results:  Type  trix (m,l)  Detected:  Error #	lisamxn lisamyn lisamy	single precision is single precision in singl	matrix, m,n,l matrix m,n,l  20.2 + 18.0n)),  Passed Oth element Oth element  Passed Oth element	Unit:

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

## MM6SN

# Input Arguments (Con't):

Type	Precision	<u>Ho</u>	ow Passed	Units
Integer (m) Integer (n) Integer (l)	SP		R5 R6 R7	- -

				MM6S3	<b>.</b> .
H2	L/S-FC LIBRARY	ROUTINE D	ESCRIPTION		
Source Member Name:	MM6S3	Size of	Code Area: _	24	Hw
Stack Requirement: _	O Hw	Data CSE	CT Size:	0	Hw
☑ Intrins	ic	☐ Pr	rocedure		
Other Library Module	es Referenced:	None.			and the second s
ENTRY POINT DESCRIPT	'IONS				
Primary Entry Name:	MM6S3				
Function: Multipl	ies two 3 x 3	single prec	ision matric	ces.	
Invoked by:					
X Compiler emitte	d code for HA	S constru	ct of the fo	orm:	
Execution Time (micr	roseconds): 4	109.6			
Input Arguments: Type	Precision		How Pass	ha:	Units
Matrix (M1)	SP		$R2 \rightarrow 0$ <sup>th</sup>	<del></del>	-
Matrix (M2)	SP		$R3 \rightarrow 0$ th		
Output Results:					
Type	Precision		How Pass		Units
Matrix (3,3)	SP		Rl → O <sup>th</sup>	element	
Errors Detected:					
Error #		Cause		Fixup	
Comments: Registers	Unsafe Across	Call: R1 F	R2.R3.R4.R5	R6.R7.F0 F1 I	72.F3 F4 T
			/ / / 100 /	/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /	-111-
Algorithm: Same as					

RETAXEDUCTULATY OF THE ORIGINAL PAGE 13 JUOR

		M(21DV	
		MM11DN	<del>'''</del>
н	AL/S-FC LIBRARY	ROUTINE DESCRIPTION	
Source Member Name:	MM11DN	Size of Code Area: 16	Hw
Stack Requirement:	O Hw	Data CSECT Size:	O Hw
🔀 Intrin	sic	☐ Procedure	
Other Library Modul	es Referenced:		
ENTRY POINT DESCRIP	TIONS		
Primary Entry Name:	MMllDN		
Function:			
Transposes an n :	x m double preci	sion matrix.	
Invoked by:	ed code for HAL/	'S construct of the form:	
or MT	where M is an n	x m double precision matrix, a	na m ana/or i
Other Library	Modules:		
Execution Time (mic	roseconds): 8.	0 + m(5.8 + 12.2n)	
Input Arguments:			••••
Type	Precision	How Passed	Units
Matrix (n,m)	DP	$R2 \rightarrow 0$ th element	
Integer Integer	SP SP	R5 R6	
Output Results:			Units
Type	Precision	How Passed	Unites
Matrix (m,n)	DP	$R1 \rightarrow 0$ <sup>th</sup> element	
Errors Detected:		Cause Fixu	n
Error #		Cause	
			ng sa Salah da Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabupatèn Kabup Kabupatèn Kabupatèn
Comments:			
Register	s Unsafe Across	Call: R1,R2,R3,R4,R5,R6,R7,F0,	F1,F2,F3.
ou de la companya de	s two nested loop iter loop selecte	ed column of input matrix;	
	ner Loop moves of corresponding ro	elements of selected column to by of result matrix.	

5-148

		_ <u>M</u> l	M11D3	
	AL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	MM11D3	Size of Code Area:	22	Hw
Stack Requirement:	O Hw	Data CSECT Size:	٥.	Hw
🛚 Intrin	sic	☐ Procedure		
Other Library Module	es Referenced:	None		
ENTRY POINT DESCRIP	PIONS			
Primary Entry Name:	MM11D3			
Function: Perfo	rms transpose o	f 3x3 double precision	matrix.	
Invoked by:				
		/S construct of the for		
TRANSPOSE (M)	where M is a 3x	3 double precision matr	ix.	
_or M				
Other Library	Modules:			
Execution Time (mici	coseconds): 9			
Taxania Barramania da				
Input Arguments: Type	Precision	How Passe	.d	Units
matrix (3,3)	DP	$R2 \rightarrow 0^{th} el$		Onr CS
ALLES BALLES				
Output Results: Type	Precision	How Passe	đ	Units
matrix (3,3)	DP	$R1 \rightarrow 0^{th} el$	CEP-P	011772
Errors Detected: Error #		Cause	Fixup	
			<u>F.I.KUP</u>	
Comments:				
	ife Across Call:	R1,R2,R4,R5,F0,F1,F2,	F3,F4,F5.	
		ements of one column in	to registers	
#-h	ctoro into we	elements of resultant fo		

<del>,</del>				MM11SN	
	•	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	Source Member Na	me: MMllsN	Size of Code Area:	16	Hw
	Stack Requiremen	t: <u> </u>	Data CSECT Size:	0	Hw
		rinsic dules Referenced:	Procedure None		
	ENTRY POINT DESC	RIPTIONS			
	Primary Entry Na	me: MM11SN			
	Function: Transposes	an n x m single pr	recision matrix.		
	TRANSPOSE (M *T or M Other Libra	, where M is and m and m and m	$\sqrt{S}$ construct of the n x m single precisor n $\neq$ 3.		
	Execution Time	(microseconds): 8	.4 + m(5.8 + 9.4n)		
	Input Arguments Type Matrix (n, Integer(# Integer (#	Decaicion	F	assed En element R5	Units
	Output Results:	Precision	How Pa	assed	Units
	<u>Type</u> Matrix(m,			o <sup>th</sup> element	
	Errors Detected Error #		Cause	Fixup	
	Comments: Re	gisters Unsafe Acro	ss Call: R1,R2,R3,R4	,R5,R6,R7,F0,Fl.	
	Widorreim.	es two nested loops Outer loop selects Inner loop loads ar of result.	which column of inpund stores column elem	t matrix to use ents as row eler	nents

			MM11	S3
· · · · · · · · · · · · · · · · · · ·		HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
	Source Member Na			18 Hw
			Data CSECT Size:	O Hw
	<b>∏</b> Int		☐ Procedure	
	Other Library Mo	dules Referenced:	None	
	ENTRY POINT DESC	RIPTIONS		
	Primary Entry Na	me: MMlls3		
	Function: Perform	s transpose of 3x3	single precision matrix.	
	or $M$		/S construct of the form	
	Execution Time	(microseconds): 71.		
	Input Arguments: Type Matrix	Precision SP	How Passed R2 $\rightarrow$ 0th elem	
	Output Results: Type Matrix	Precision SP	How Passed Rl $\rightarrow$ 0th elem	
	Errors Detected Error #		Cause	Fixup
	Comments: Regis	ters Unsafe Across	Call: R1,R2,R4,R5,F0,F1,	F2,F3.
	Algorithm:	Uses loop to load matrix and store to columns 1 → 3, row	F0, F2, F4 with columns of them as rows of output may $1 \rightarrow 3$ .	of input trix for

REPRODUCIBILITY OF THE

		MM	12DN	-
HAL/S-FC LIBRAR	RY ROUTINE DESCRI	PTION		
Source Member Name: MM12DN	Size of Code	Area:	150	Hv
Stack Requirement: 22 Hw	Data CSECT Si		0	Hw
☐ Intrinsic	X Procedu			***
Other Library Modules Referenced:	None.			
ENTRY POINT DESCRIPTIONS				
Primary Entry Name: MM12DN				
· · · · · · · · · · · · · · · · · · ·				
Function: Find the determinant of	an n x n double	precision	matrix.	
Invoked by:				
X Compiler emitted code for HAI	L/S construct of	the form		
D = DET(M), where D is a dec	lared double non	oidian -	•	
M an n x n	, double precisi	on matric	tar, and	
Other Library Modules:	ETACTOT	ou macrix,		
ting the second of the second				
Execution Time (microseconds): fo	$or n = 2 \cdot 62 \cdot 2$			
	or n > 4. //~~~~	The said		
Input Argument	or $n \geq 4$ : (Conti	nued on bo	ttom of t	his p
input Arguments:	or n <u>&gt;</u> 4: (Conti	nued on bo	ttom of t	his p
Type Precision		nued on bo	ttom of t	
Type Precision  Matrix(n,n)	<u> Ho</u>	w Passed		his p <u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP	<u> Ho</u>			
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP	<u> Ho</u>	w Passed $\rightarrow$ 0 <sup>th</sup> elem		
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:	<u>Ho</u> R2	w Passed → O <sup>th</sup> elem R4 R5		
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision	<u>Ho</u> R2	w Passed → O <sup>th</sup> elem		<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:	<u>Ho</u> R2	w Passed → O <sup>th</sup> elem R4 R5		<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision	<u>Ho</u> R2	w Passed  → 0 <sup>th</sup> elem R4 R5 w Passed		<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP	Ho	w Passed  → 0 <sup>th</sup> elem R4 R5 w Passed	ent	<u>U</u> n
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:	<u>Ho</u> R2	w Passed  → 0 <sup>th</sup> elem R4 R5 w Passed		<u>U</u> n
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #	Ho	w Passed  → 0 <sup>th</sup> elem R4 R5 w Passed	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None	Ho	w Passed  → 0 <sup>th</sup> elem R4 R5 w Passed	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>U</u> n
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>U</u> n
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call  Algorithm:  Same as MM12SN	Ho R2 Ho	w Passed → 0 <sup>th</sup> elem R4 R5 w Passed F0	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected: Error #  None  Comments: Registers Unsafe Across Call  Algorithm: Same as MM12SN  Execution Time (Continued):	Ho R2  Ho Cause  1: F0,F1,F2,F3,F	w Passed  → 0th elem  R4  R5  w Passed  F0	ent	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call  Algorithm:  Same as MM12SN  Execution Time (Continued):  minimum time = 59.4 + 10.2n <sup>2</sup> + \(\Sigma\) k=1	Ho R2  Ho Cause  1: F0,F1,F2,F3,F  (54.8k <sup>2</sup> + 81.2k	w Passed  + Oth elem R4 R5  w Passed F0  '4,F5.  + 115.6)	Fixup	
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected:  Error #  None  Comments:  Registers Unsafe Across Call  Algorithm:  Same as MM12SN  Execution Time (Continued):  minimum time = 59.4 + 10.2n <sup>2</sup> + \(\Sigma\) k=1	Ho R2  Ho Cause  1: F0,F1,F2,F3,F  (54.8k <sup>2</sup> + 81.2k	w Passed  + Oth elem R4 R5  w Passed F0  '4,F5.  + 115.6)	Fixup	<u>Un</u>
Type Precision  Matrix(n,n) DP  Matrix(n,n) workarea DP  Integer(n) SP  Output Results:  Type Precision  Scalar DP  Errors Detected: Error #  None  Comments: Registers Unsafe Across Call  Algorithm: Same as MM12SN  Execution Time (Continued):	Ho  R2  Ho  Cause  1: $F0,F1,F2,F3,F$ $(54.8k^2 + 81.2k^2)$ $(60.2k^2 + 134.8)$	w Passed  + Oth elem R4 R5  w Passed F0  '4,F5.  + 115.6)	Fixup	<u>Un</u>

5-152

		-	MM12D3	
АН	//S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	MM12D3	Size of Code Area:	44	Hw .
Stack Requirement: _	18 Hw	Data CSECT Size:	0	_ Hw
☐ Intrins	ic	X Procedure		
Other Library Module	s Referenced:	None.		
ENTRY POINT DESCRIPT	TONS			
Primary Entry Name:	MM12D3			
Function:		precision matrix.		
Invoked by:  X Compiler emitte	ed code for HA	L/S construct of the	form:	
DET(M), where	M is a double	precision 3x3 matrix.		
X Other Library	Modules: Mi	114D3		
Execution Time (mic	roseconds):	229.6		
Input Arguments:	Precision	How Pa	ıssed	Units
Type Matrix (3,3)	DP	R2 → O <sup>t</sup>	h element	
Output Results:	Precision	How Pa	issed	Units
<u>Type</u> Scalar	DP	FC		
Errors Detected: Error #		<u>Cause</u>	Fixup	
Comments: Registers Un	safe Across Ca	ll: F0,F1,F2,F3,F4,F5		
Algorithm: Uses	direct code, r	no loops to calculate	determinant.	

도함 열리 시간 기가 가장 그 보는 보고 있는데 기가 그 가 기계로 가는 그리고 되었다.

			, <u>M</u>	M12SN	
HAL/	S-FC LIBRARY	ROUTINE DES	CRIPTION		
	MM12SN			1308	Men
Source Member Name:	FINEZON		de Area:		. Hw
Stack Requirement:	20 Hw	Data CSECT	Size:	0	Hw
☐ Intrinsic		v Proc	edure		
Other Library Modules	Referenced:	None.			
ENTRY POINT DESCRIPTION	NS				
Primary Entry Name:	MM12SN				
Function: Finds the	determinant	of an nxns	ingle preci	sion matrix.	
Invoked by:  X Compiler emitted	code for HAI	L/S construct	of the for	m:	
S = DET(M); wh		lared a singl			
Other Library Mod	ules:				
Execution Time (micros		or $n = 2: 44.6$ or $n \ge 4: (Con$		next page)	
Input Arguments:					•
<u>Type</u>	Precision		How Passe R2 → 0th		Units
Matrix Matrix(n,n) workar Integer(n)	SP ea SP SP		R2 7 0 0 12 R4 R5	etellieur	
Output Results:	<b>3</b> F		, KS		
Type	Precision		How Passe	e <b>d</b>	Units
Scalar	SP		FO		
Errors Detected:					
Error #		Cause		Fixup	
None					
				Tariharina arrangan merenakan	
Comments: Registers Unsa	afe Across Ca	all: FO,F1,F	2,F3,F4,F5.	Circus	
. Algorithm: DET = 1.					
FOR K = BIG =	1 TO N1 DO				
	J1 = K				
	= K TO N DO		<b>.</b>	nd maximal ele	ment
	= K TO N D			in maximat ete	emeti r
	ABS(A(I,J))		~		
	BIG = ABS(A) $Il = I;$	(I,J));			
	1				
되어 하는 물건이 조리를 받았다. ^^	J1 = J;		er Jarejela		

(Continued on next page)

MM12SN

MM12SN

Execution Time (microseconds) (Continued):

minimum time = 
$$47.8 + 7.8n^2 + \sum_{k=1}^{n-1} (37.6k^2 + 64.6k + 85.8)$$

maximum time = 
$$47.8 + 7.8n^2 + \sum_{k=1}^{n-1} (41.6k^2 + 105.8k + 127.2 + 3.6n)$$

The minimum time occurs in the event that all matrix elements are positive and where no row or column switching is required at any point of the computation.

The maximum time occurs in the event that all matrix elements require complementing to obtain their absolute value, BIG changes on every comparison, and row and column switching are required at every point in the computation.

Algorithm (Con't)

DET = -DET

FOR J = K TO N SWITCH(A(I1,J),A(K,J) switch rows

IF J1  $\neq$  K THEN

DET = -DET

FOR I = K TO N SWITCH(A(I,J1),A(JK); switch columns.

DET = DET\*A(K,K) product of diagonal element

FOR I = K + 1 TO N DO

TEMP1 = -A(1,K)/A(K,K) reduce

FOR J = K + 1 TO N DO

A(I,J) = A(I,J) + A(K,J) \* TEMP1

DET = DET\*A(N,N) last diagonal element

If dim = 2, then special case:

DET = A(1,1) \*A(2,2) -A(1,2) \*A(2,1)

	MM12S3	
HAL/S-FC LIBRARY ROUTINE DESCRIPTION		
Source Member Name:MM12S3 Size of Code Area:		Hw
Stack Requirement: 18 Hw Data CSECT Size:		Hw
Intrinsic M Procedure		
Other Library Modules Referenced: None.		
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: MM12S3		
Function: Finds determinant of a single precision 3x3 matrix.		
Invoked by:  [X] Compiler emitted code for HAL/S construct of the DET(M), where M is a 3x3 single precision matrix.		
X Other Library Modules: MM14S3		
Execution Time (microseconds): 116.0		
Input Arguments:  Type Precision How Pa		<u>Units</u>
Matrix(3,3) SP $R2 \rightarrow 0^{th}$	element	
Output Results:  Type Precision How Page 1	assed	Units
Errors Detected: Error # Cause	Fixup	
Comments: Registers Unsafe Across Call: F0,F1,F:	2,F3,F4,F5.	
Algorithm: Uses direct inline code to calculate		
$det = M_{11} M_{22} M_{33} + M_{12} M_{23} M_{31} + M_{13} M_{22}$	21 <sup>M</sup> 32	

			MM13DN	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Na		Size of Code Area: _	10	Hw
Stack Requiremen		Data CSECT Size:	. 0	Hw
		Procedure	ESCRIPTION  Code Area: 10 Hw  CT Size: 0 Hw  cocedure  ecision matrix.  act of the form: cision matrix, n=3.	
<del></del>	rinsic	None		
Other Library Mo	dules Referenced:	None		
ENTRY POINT DESC	CRIPTIONS			
Primary Entry Na	me: MM13DN			
Function: Calculates	TRACE of an n x n	double precision matr	i. i.e., - 1939 - 197	
Invoked by:  Z Compiler en	nitted code for HAI	L/S construct of the f	orm:	
TRACE (M), wh	nere M is an n x n	double precision matri	x, n≠3.	
Other Libra	ary Modules:			
				0
Execution Time	(microseconds):	12.0 + 10.2n		
Input Arguments Type	: Precision	How Pas	sed	Units
Matrix (n,n)	DP		element	•
Integer (n)	SP	y a dia na managara a <b>Ro</b> a Tanangara		
Output Results:	Precision	How Pas	sed	Units
Scalar	DP	FO		
Errors Detected Error #		<u>Cause</u>	Fixup	
Comments: Registers	s Unsafe Across Cal	1: R2,R4,R5,R6,F0,F1.		
Algorithm:	to zero, each pas	lue of n+1; Uses loop s summing a diagonal e value to increment fr	lement of the	matrix

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

		MMl	3D3	
на	L/S-FC LIBRARY ROUTIN	NE DESCRIPTION		
Source Member Name:			8	Hw
Stack Requirement:		CSECT Size:	0	Hw
X Intrins		Decadure	•	
Other Library Module		e		· .
ENTRY POINT DESCRIPT				
Primary Entry Name:	WWT3D3			
Function:	of a 3x3 double prec	ision matrix.		
		andre de la companya de la companya Companya de la companya de la compa		
Invoked by:  Compiler emitte	ed code for HAL/S cor	astruct of the form	1:	
	M is a 3x3 double pr			
Other Library N				
Other Hibrary	locares.			
Execution Time (mic	roseconds): 19.8			
Input Arguments:	Precision	How Passe	<u>a</u>	Uni
Type Matrix (3,3)	DP	$R2 \rightarrow 0$ <sup>th</sup> ele	ement	
Macrix (5,5)				
Output Results:	Precision	How Passe	<u>d</u>	Uni
<u>Type</u> Scalar	DP	FO		-
Errors Detected:	요요 가능하루스 수 있는 것이다. 사람들은 기계를 하는 것이다.			
Error #	<u>Cav</u>	ise	Fixup	
Comments:	afe Across Call: R2,	R4,F0,F1.		
	1	calculate		
	ct code, no loops to	calculate		

				MM13SN		
	ІАН	J/S-FC LIBRARY	ROUTINE DESCRIPT	TION		
	Source Member Name:	MMl3SN	Size of Code Ar	rea: 8	·	Hw
	Stack Requirement:	0 Hw	Data CSECT Size	):	0	_ Hw
	X Intrins		☐ Procedur	<b>e</b>		
	Other Library Module	s Referenced: _	None.			
<del></del>	ENTRY POINT DESCRIPT	IONS				
	Primary Entry Name:	MM13SN				
	Function: Calculates TRACE (	of an n x n sin	gle precision ma	trix.		
	Invoked by:  X Compiler emitte  TRACE (M), where		/S construct of precision n x n		3.	
	Other Library M	odules:				
	Execution Time (micr	roseconds): 8	.8 + 6.2n			
	Input Arguments: Type	Precision	<u>Hc</u>	w Passed		Units
	Matrix (n,n) Integer(n)	SP SP	R2 →	O <sup>th</sup> element R5		
	Output Results: Type	Precision	Hc	w Passed		Units
	<u>rype</u> Scalar	SP		FO		_
	Errors Detected: Error #		<u>Cause</u>		Fixup	
	Comments: Registers	Unsafe Across (	Call: R2,R4,R5,	R6,F0,Fl.		

Algorithm:

Creates a skip value of n+1; uses loop counting down n-1 to zero, each pass summing a diagonal element of the matrix by using the skip value to increment from the previous diagonal element.

HAL/S-FC LIBRARY ROUTINE DESCRIPTIONS  Primary Entry Name: MM13S3  Function:  Calculates TRACE of a 3x3 single precision  Invoked by:  Compiler emitted code for HAL/S construction:	ode Area:	Hw Hw
Source Member Name: MM13S3 Size of Constact Requirement: MM13S3 Size of Constact Requirement: MM13S3 Process Mone.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision Invoked by:	ode Area:	
Stack Requirement:    O	T Size: 0.	
Other Library Modules Referenced: None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision Invoked by:	ocedure	Hw
Other Library Modules Referenced: None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision Invoked by:		•
ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision Invoked by:	matrix.	
ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision Invoked by:	matrix.	
Primary Entry Name: MM13S3  Function: Calculates TRACE of a 3x3 single precision  Invoked by:	matrix.	
Function:  Calculates TRACE of a 3x3 single precision  Invoked by:	matrix.	
Calculates TRACE of a 3x3 single precision Invoked by:	matrix.	
	•	
X compiter emirced code for many conserve	t of the form:	
TRACE(M), where M is a 3x3 single preci	ision matrix.	
Other Library Modules:		
Execution Time (microseconds): 9.8		
Input Arguments:		
Type Precision	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$	Unit
Matrix (3,3) SP	R2 7 0 m element	
Output Results:		
<u>Type</u> <u>Precision</u> Scalar SP	How Passed FO	Unit
Errors Detected:		
Error # Cause	Fixup	

		Procedure  MM15DN   double precision matrix.  or HAL/S construct of the form: M is an n x n double precision matrix, n ≠ 3.  ): for n = 2: 173.8,     for n ≥ 4: 63.0 + 129.5n + 43.0n <sup>2</sup> + 65.4n <sup>3</sup> sion  How Passed R4 → 0th element R5 R7  sion How Passed R2 → 0th element R2 Singular matrix Return identity matrix		
			\ <u></u>	
	HAL/S-FC LIB	RARY ROUTINE DESC	CRIPTION	
Sour	ce Member Name: MM14D	N Size of Coo	de Area: 2	58 Hw
Stac	k Requirement: 20	Hw Data CSECT	Size:	Hw
	☐ Intrinsic	Y Proc	edure	
Othe	r Library Modules Referenc	ed: MM15DN		
ENTR	Y POINT DESCRIPTIONS			
Prim	ary Entry Name: MM14D	N		
Func	tion: Inverts an n x n d	ouble precision r	matrix.	
X	ked by:  Compiler emitted code for  M , where M  or INVERSE(M)	HAL/S construct is an n x n doub	of the form: le precision mat	erix, n ≠ 3.
	Other Library Modules:	for $n = 2 \cdot 173$		
Бхес	defour time (wreteresequing).			$0.0n^2 + 65.4n^3$
Inpu	t Arguments:			
	Type Precisi	<u>on</u>		
	Matrix(n,n) DP Integer(n) SP Matrix(n,n) workarea DP		R5	3nc
Out	out Results:		How Daggod	Inits
	<u>Type</u> <u>Precisi</u> Matrix DP	<u>.on</u>		
Erro	ors Detected:			
	Error # 27			
Comm	nents: Registers Unsafe A	Across Call: F0,E	F1,F2,F3,F4,F5.	
Algo	orithm: Same as MMl4SN, ex is done by multipl use of long divide	Lying by reciproc	element divide al to some time	operation over

			M	414D3	<b>.</b>
	HAL/S-FC LIBRAR	Y ROUTINE DESC	RIPTION		
Source Member Nam	me:MM14D3	Size of Code	e Area:	128	Hw
Stack Requirement	t: <u>18</u> Hw	Data CSECT	Size:	2	Hw
☐ Int:	rinsic	Proc	edure		
Other Library Moo	dules Referenced:	MM12D3	MM15DN		
ENTRY POINT DESC	RIPTIONS				
Primary Entry Na	me: MM14D3		R	EPRODUCI RIGINAL I	BILITY OF 'E PAGE IS POO
Function:	s a 3 <b>x</b> 3 double pi	recision matrix			
	•				
*	itted code for HA				
M -  or INVERSE(M)	, where M is a 3x	k3 double preci	sion matri	х.	
Other Libra					
Execution Time (	microseconds):	795.4			
		795.4			
		795.4	$\frac{\text{How Passe}}{\text{R4} \to 0^{\text{th}}}$		<u>Units</u>
Input Arguments: <u>Type</u> Matrix(3,3)	Precision	795.4			<u>Units</u>
Input Arguments:  Type Matrix(3,3)  Output Results:	Precision			element	Units - Units
Input Arguments: <u>Type</u> Matrix(3,3)	Precision DP		R4 → Oth	element	
Input Arguments:  Type Matrix(3,3)  Output Results:  Type Matrix(3,3)  Errors Detected:	Precision DP Precision		R4 + 0th	element <u>d</u> element	
Input Arguments:  Type Matrix(3,3)  Output Results: Type Matrix(3,3)	Precision DP Precision	Cause	R4 $\rightarrow$ 0th  How Passe R2 $\rightarrow$ 0th	element  d element  Fixup	Units
Input Arguments:  Type Matrix(3,3)  Output Results: Type Matrix(3,3)  Errors Detected: Error # 27	Precision DP  Precision DP	<u>Cause</u> rse of singular	$R4 \rightarrow 0$ th  How Passe $R2 \rightarrow 0$ th  matrix.	element d element <u>Fixup</u> Re <b>bur</b> n ider	Units

			MMl	1SN	
	HAL/S-FC LIBRARY	ROUTINE DESCR	IPTION		
Source Member Name		Size of Code		242	Hw
			O	2	Hw
Stack Requirement:	20 Hw				nw
☐ Intri	nsic	Procedu	re		
Other Library Modu	les Referenced:	MM15SN			
ENTRY POINT DESCRI	PTIONS				
Primary Entry Name	: MMl4SN				
Function: Inverts		sion n x n matr	ix.		
Invoked by:  X Compiler emit					
or INVERSE(M)	, where M is a s	ingle precision	n x n mat	rix, $n \neq 3$	3.
Of INVERSE (II)  Other Library					
Execution Time (mi	croseconds): for	n = 2: 107.6,		•	3
	for	n = 4: 52.0	+ 39.2n + 1	LO.5n° + 5	64.6n
Input Arguments:	Precision		How Passed		Units
Input Arguments: <u>Type</u> Matrix(n,n)	Precision SP		How Passed 4 → 0 <sup>th</sup> el	ement	<u>Units</u>
<u>Type</u> Matrix(n,n) Integer(n)	SP SP		$4 \rightarrow 0^{ ext{th}} \text{ el}$ R5	ement	Units -
Type  Matrix(n,n)  Integer(n)  Matrix(n,n) wo	SP SP		$4 \rightarrow 0^{\text{th}} \text{ el}$	ement	Units
<u>Type</u> Matrix(n,n) Integer(n)	SP SP		$4 \rightarrow 0^{ ext{th}} \text{ el}$ R5		Units - - Units
Type  Matrix(n,n)  Integer(n)  Matrix(n,n) wo  Output Results:	SP SP orkarea SP	<b>.</b>	4 → 0 <sup>th</sup> el R5 R7		
Type  Matrix(n,n)  Integer(n)  Matrix(n,n) wo  Output Results:  Type	SP SP orkarea SP Precision	<b>.</b>	$4 \rightarrow 0$ th el R5 R7 How Passed	ement	
Type  Matrix(n,n)  Integer(n)  Matrix(n,n) wo  Output Results:  Type  Matrix	SP SP orkarea SP Precision	<b>.</b>	4  ightharpoonup 0th el R5 R7 How Passed 2  ightharpoonup 0th el	ement <u>Fixup</u>	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected:	SP SP orkarea SP <u>Precision</u> SP	F	4  ightharpoonup 0th el R5 R7 How Passed 2  ightharpoonup 0th el	ement	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments:	SP SP orkarea SP <u>Precision</u> SP Matrix i	F <u>Cause</u> s singular.	4  ightarrow 0th el R5 R7 How Passed 2  ightarrow 0th el	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments:	SP SP orkarea SP <u>Precision</u> SP	F <u>Cause</u> s singular.	4  ightarrow 0th el R5 R7 How Passed 2  ightarrow 0th el	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register	SP SP orkarea SP  Precision SP  Matrix i	F <u>Cause</u> s singular.	4  ightarrow 0th el R5 R7 How Passed 2  ightarrow 0th el	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register  Algorithm: For K FI	SP SP Orkarea SP  Precision SP  Matrix i SS Unsafe Across  I, N ND MAXIMAL ELEME	Cause s singular.  Call: F0,F1,F	$4 \rightarrow 0$ th el R5 R7 How Passed $2 \rightarrow 0$ th el Return i	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register Algorithm: For K FI	SP SP SP orkarea SP  Precision SP  Matrix i TS Unsafe Across = 1, N ND MAXIMAL ELEme we it as 'big' p	Exause  S singular.  Call: F0,F1,F  nt in row K to ivot element	$4 \rightarrow 0$ th el R5 R7 How Passed $2 \rightarrow 0$ th el Return i	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register  Algorithm: For K FI sai	SP SP orkarea SP  Precision SP  Matrix i TS Unsafe Across  = 1, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K)	$4 \rightarrow 0$ th el R5 R7 How Passed $2 \rightarrow 0$ th el Return i	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register Algorithm: For K FI sai	SP SP Orkarea SP  Precision SP  Matrix i TS Unsafe Across  = 1, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K)	$4 \rightarrow 0$ th el R5 R7 How Passed $2 \rightarrow 0$ th el Return i	ement <u>Fixup</u> dentity ma	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register Algorithm: For K FI sa' sa' sa' sw' sw'	SP SP SP Orkarea SP  Precision  SP  Matrix i  SS Unsafe Across  I, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as itch Kth and ISW itch Kth and JSW	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K) (K) th row (K) th col	4 → 0th el R5 R7 How Passed 2 → 0 <sup>th</sup> el Return i 2,F3,F4,F5 n, cols, K	ement  Fixup  dentity many	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register Algorithm: For K FI sa sa sa sw sw sw di	SP SP orkarea SP  Precision SP  Matrix i  SS Unsafe Across  I, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as itch Kth and ISW itch Kth and JSW vide Kth col exc	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K) (K) th row (K) th col	4 → 0th el R5 R7 How Passed 2 → 0 <sup>th</sup> el Return i 2,F3,F4,F5 n, cols, K	ement  Fixup  dentity many	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register Algorithm: For K FI sa sa sa sw sw sw di re	SP SP Orkarea SP  Precision  SP  Matrix i  SS Unsafe Across  1, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as itch Kth and ISW itch Kth and JSW vide Kth col exc duce matrix	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K) (K)th row (K)th col ept for Kth element	4 → 0th el R5 R7 How Passed 2 → 0 <sup>th</sup> el Return i 2,F3,F4,F5 n, cols. K	ement  Fixup  dentity many  to n	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register  Algorithm: For K FI sa sa sw sw di re di re	SP SP orkarea SP  Precision  SP  Matrix i  SS Unsafe Across  = 1, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as itch Kth and ISW itch Kth and JSW vide Kth col exc duce matrix vide Kth row exc place pivot by r	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K) (K)th row (K)th col ept for Kth ele eciprocal	4 → 0th el R5 R7 How Passed 2 → 0 <sup>th</sup> el Return i 2,F3,F4,F5 n, cols, K	ement  Fixup  dentity many  to n	Units
Type Matrix(n,n) Integer(n) Matrix(n,n) wo Output Results: Type  Matrix Errors Detected: Error #  27  Comments: Register  Algorithm: For K FI sa sa sw sw di re di re	SP SP Orkarea SP  Precision  SP  Matrix i  SS Unsafe Across  1, N ND MAXIMAL ELEme ve it as 'big' p ve its row # as ve its col # as itch Kth and ISW itch Kth and JSW vide Kth col exc duce matrix	Cause s singular.  Call: F0,F1,F  nt in row K to ivot element ISW(K) JSW(K) (K)th row (K)th col ept for Kth ele eciprocal	4 → 0th el R5 R7 How Passed 2 → 0 <sup>th</sup> el Return i 2,F3,F4,F5 n, cols, K	ement  Fixup  dentity many  to n	Units

	WI 402	
	MM14S3	
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION	1
	Source Member Name: MM14S3 Size of Code Area: 80	Hw
	Stack Requirement: 18 Hw Data CSECT Size: 2	Hw
	☐ Intrinsic ☑ Procedure	
	Other Library Modules Referenced: MM12S3, MM15SN	
	ENTRY POINT DESCRIPTIONS	
	Primary Entry Name: MM14S3	
	Function: Inverts a 3x3 single precision matrix.	
	Invoked by:	
	Compiler emitted code for HAL/S construct of the form:	
	$^{ ext{M}}$ , where M is a 3x3 single precision matrix. or INVERSE(M)	
	Other Library Modules:	
	Execution Time (microseconds): 458.8	
	Input Arguments:	
	Type Precision How Passed	Units
1000	Matrix SP R4 + 0 <sup>th</sup> element	-
	Output Results:	
	Type Precision How Passed	Units
	Matrix SP R2 $\rightarrow$ 0 <sup>th</sup> element	-
	Errors Detected:  Error # Cause Fixup	
	27 Attempted inverse of singular matrix. Return an id	entity
	matrix.	<b></b>
	Comments:  Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.	
	Algorithm: Explicit code, no loops to calculate:	
	inverse $M = \frac{\text{adj}(M)}{ M }$ , where $\text{adjM}_{i,j} = \text{detM}_{i\neq 3,j\neq 3}$ and $ M  = \text{detM}$	
	uses external determinant routine (MM12S3) and in event of determi of zero, calls identity matrix routine (MM15SN).	nant

					1
				MM.	L5DN
	HAL/S-FC	LIBRARY I	ROUTINE DESCRI	PTION	
HAL/S-FC LIBRARY ROUTINE DESCRIPTION  Source Member Name: MM15DN Size of Code Area: 18  Stack Requirement: 0 Hw Data CSECT Size: 0  [Intrinsic Procedure]  Other Library Modules Referenced: None  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MM15DN  Function:  Creates an n x n double precision identity matrix.  Invoked by:  [X] Compiler emitted code for HAL/S construct of the form: *0 , where M is an n x n double precision matrix.  [X] Other Library Modules: MM14DN, MM14D3  Execution Time (microseconds): 15.6 + 5.0n + 11.2n <sup>2</sup> Input Arguments:  Type Precision How Passed Integer (n) SP R5  Output Results:  Type Precision How Passed Matrix(n,n) DP R1 + 0th element Errors Detected:	Source Member Name:	M15DN	Size of Code	Area: <u>18</u>	Hw
	Hw				
	Stack Requirement:    O   Hw   Data CSECT Size:   O   E				
	Other Library Modules Refe	renced:	None		
	ENTRY POINT DESCRIPTIONS				
	Primary Entry Name: MM15	DN			
	Function:				
	Creates an n x n double p	recision	identity matri	<b>Lx.</b>	
	Compiler emitted code				
	X Other Library Modules	: MM14D	N, MM14D3		
	Execution Time (microsecon	ds): 15	5.6 + 5.0n + 11	L.2n <sup>2</sup>	
	Type Pre		<u> </u>	<del></del>	Units
	<u>Type</u> Pre				<u>Units</u>
			Cause	<u>F</u>	ixup
		cross Cal	.l: Rl.R4.R5.R6	5.R7.F0.F1.F2.	F3
e filozofi i lengdyn Holloffi i lengdyn					[14] [14] 12] 14] 14] 16] 16] 16] 16] 16] 16] 16] 16] 16] 16] 16] 16] 16]
	Inner loop	compares at currer	, each counting both loop ind nt row/column p	ices; if equa	

	MM15SN						
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION						
	Source Member Name: MM15SN Size of Code Area: 14	Hw .					
	Stack Requirement: 0 Hw Data CSECT Size: 0	Hw .					
	X Intrinsic Procedure						
	Other Library Modules Referenced:						
<del> </del>	ENTRY POINT DESCRIPTIONS	•					
	Primary Entry Name: MM15SN						
	Function:  Creates an n x m identity matrix.						
	Invoked by:						
	Compiler emitted code for HAL/S construct of the form:  **O  **O  **O  **  **  **  **  **  **						
	X Other Library Modules: MM14SN, MM14S3						
	Execution Time (microseconds): 10.0 + 5.2n + 9.6n <sup>2</sup>						
	Input Arguments:  Type Precision How Passed  Integer (n) SP R5	Units -					
	Output Results:  Type Precision How Passed  Matrix $(n,n)$ SP $Rl \rightarrow 0$ $th$ element	<u>Units</u>					
	Errors Detected: Error # Cause Fix	ip					
	Comments:  Registers Unsafe Across Call: Rl,R4,R5,R6,R7,F0,F1,F2,F3.						
	Algorithm: Uses two nested loops, each counting 1 to n.  Inner loop checks both loop indices; if equal, sto a 1.0 at current row/column position, otherwise st	res ores 0.0,					



				MM17D	3	
	LYNT	/c_FC T.TRRARY	ROUTINE DESCRI	PTION		
					6 Hw	P
	Member Name:					
Stack 1	Requirement:	20 Hw	Data CSECT Si	.ze:	0 . Hw	
	☐ Intrinsi	c	Procedu	re .		
Other	Library Modules	Referenced:	None			_
ENTRY	POINT DESCRIPTI	ONS				
	y Entry Name:					
			• • • • • • •	in to a nowe	r	
Functi	on: Raises a	3 X 3 GOGDIC	Processia			
	* 1					
Invoke	d by:	7.	/e construct o	f the form:		
Lxl C	ompiler emitted	. code for har	1/2 COHSCIACE O			
	M, where M is	a 3x3 double	precision matr:	ix, and		
	Iis	an integer li	teral $\geq 1$ .			
П.						
, LJ <sup>0</sup>	ther Library Mo	dures.				
	ion Time (micro	· ( abrona	(On next page	).		
Execu	tion Time (MICIC	)S3COnd5/.				
Thout	Arguments:			·		T7
11.500	Туре	Precision		How Passed		Un
	natrix(3,3)	DP.	R4	→ 0 <sup>th</sup> elemen	it	
	integer(power)			R6		
<b>1</b>	latrix(3,3) work			R7		
Outpu	t Results:	Precision		How Passed		Ur
	<u>Type</u>		מת	→ Oth elemen	nt	-
i	Matrix (3,3)	DP	K2	0 670		
	w. b					
Error	s Detected:		Cause		Fixup	
	Error #		<u>Cause</u>			
		Imaafa Xar	oss Call: FO,F	1.F2.F3.F4.F5	5.	
Comme	nts: Registe:	rs unsale ACIO	)33 Carr. 1011			
			etaja rajabajā jūda,			
		we sin near	al 3 and drops	in MM17DN co	de.	
	ithm: Loads	KO WITH LITER	ar and aroba			

### MM17D3

Execution Time (microseconds):

Exponent = 2: 991.6

Exponent > 2: 1071.2 · (# of significant zeros in exponent)

+ 2137.2 · (# of ones in exponent)

- 2105.8

-	
MM17D3	

Secondary Entry Name: MM17DN			
Function: Raises an n x n d	ouble precisio	on matrix to a powe	er.
Invoked by:			
X Compiler emitted code for	HAL/S constru	act of the form:	
M <sup>I</sup> , where M is an n x n d I is an integer	louble precisi	on matrix, and	
Other library modules:			
Execution Time (microseconds):	(See be	elow)	
Input Arguments: Type Precisi	ion	How Passed	Units
Matrix(n,n) DP integer(n) SP integer(power) SP matrix(n,n) workarea DP		R4 → Oth elemer R5 R6 R7	it
Output Results:	ion	How Passed	Units
Matrix (n,n) DP		$R2 \rightarrow 0$ <sup>th</sup> elem	nent
Errors Detected: Error #	<u>Cause</u>		Fixup
Comments:	1. FO Fl.F2.I	3.F4,F5.	

Registers Unsafe Across

Algorithm: Same as MM17SN.

REPRODUCIBILITY OF THE ÓRIGINAL PAGE IS POOR

Execution Time (microseconds):

 $27.8n^3 + 19.4n^2 + 6.2n + 43.4$  if power = 2.

124.2 + TMULT(KA) + TMOVE(KA-1) + 8.6 KB + 3.4 KC if power > 2.

where:

TMULT =  $9.6 + 6.2n + 19.4n^2 + 27.8n^3$ 

=  $(((# significant 1's in exponent)-1)\cdot 2) + (# of significant)$ 0's in exponent)

 $TMOVE = 10.2 + 11.0n^2$ 

= total number of significant l's and 0's in exponent

= # of significant l's in exponent. KC

			MM17s3	<del></del> 7
H.	L/S-FC LIBRARY F	ROUTINE DESCRIPTION	<b>3</b>	
Source Member Name:	MM17s3	Size of Code Area:	78	Hw
Stack Requirement:	20 <b>Hw</b>	Data CSECT Size:	0	Hw
☐ Intrins	sic	X Procedure		
Other Library Module	es Referenced:	None.		
ENTRY POINT DESCRIPT	TIONS			
Primary Entry Name:	MM17s3			
Function: Raises	a 3x3 single pre	scision matrix to a	DOWER	
Marses	a ava attilate bie	CTSTOIL MIGCELTY CO d	· Power •	
Invoked by				
Invoked by:	ed code for HAL/S	construct of the	form: .	
**	ls a 3x3 single p			
/				
je je je kajeje de <b>I</b> j <b>i</b>	is an integer lit			
I i Other Library N	is an integer lit			
and the second s	is an integer lit			
Other Library	is an integer lit			
and the second s	is an integer lit	eral <u>&gt;</u> 1.		
Other Library M	is an integer lit	eral <u>&gt;</u> 1.		
Other Library	is an integer lit	eral $\geq 1$ .  on next page)  How Page	assed	
Execution Time (microstrum)  Input Arguments:  Type matrix(3,3)	is an integer lit Modules:  roseconds): (C  Precision  SP	eral $\geq 1$ .  on next page) $\frac{\text{How Poly}}{\text{R4} \rightarrow 0^{\frac{1}{2}}}$	assed <sup>h</sup> element	
Execution Time (mice Input Arguments:  Type matrix(3,3) integer(power)	is an integer lit Modules:  roseconds): (C  Precision  SP  SP	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4} \rightarrow 0^{\frac{1}{2}}}$	assed <sup>h</sup> element 6	
Execution Time (microstrum)  Input Arguments:  Type matrix(3,3) integer(power) matrix(3,3) wor	is an integer lit Modules:  roseconds): (C  Precision  SP  SP	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4} \rightarrow 0^{\frac{1}{2}}}$	assed <sup>h</sup> element	<u> </u>
Other Library M  Execution Time (micro  Input Arguments:  Type matrix(3,3) integer(power) matrix(3,3) wor  Output Results:	is an integer lit Modules:  roseconds): (C  Precision  SP  SP	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4}} \rightarrow 0^{\text{t}}$ R  How Poly	assed h element 6 7	
Execution Time (microstrum)  Input Arguments:  Type matrix(3,3) integer(power) matrix(3,3) wor	is an integer lit Modules:  roseconds): (C  Precision  SP  SP  SP  ckarea SP	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4}} \rightarrow 0^{\text{t}}$ R  How Poly	assed h element .6	
Other Library Memory Me	s an integer lit  Modules:  roseconds): (C  Precision SP SP SP karea SP  Precision	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4}} \rightarrow 0^{\text{t}}$ R  How Poly	assed h element 6 7	
Other Library M  Execution Time (micro Input Arguments:  Type matrix(3,3) integer(power) matrix(3,3) wor  Output Results: Type	s an integer lit  Modules:  roseconds): (C  Precision SP SP SP karea SP  Precision	eral $\geq$ 1.  In next page) $\frac{\text{How Poly}}{\text{R4}} \rightarrow 0^{\text{t}}$ R  How Poly	assed h element 6 7	
Input Arguments:  Type matrix(3,3) integer(power) matrix(3,3) wor  Output Results:  Type Matrix  Errors Detected:	s an integer lit  Modules:  roseconds): (C  Precision SP SP SP karea SP  Precision	eral $\geq$ 1.  In next page) $ \frac{\text{How Pr}}{\text{R4}} \rightarrow 0^{\text{t}} $ R $ \frac{\text{How Pr}}{\text{R}} $ R $ \frac{\text{How Pr}}{\text{R2}} \rightarrow 0^{\text{t}} $	assed h element 6 7 assed h element	

#### MM17S3

Execution Time (microseconds):

exponent = 2: 623.6

exponent > 2: 681.6 · (# significant zeros in exponents)

+ 1358.0 · (# ones in exponent)

- 1305.0.

MM17S3	

Secondary Entry Name: MM17SN

Function: Raises an n x n single precision matrix to a power.

Invoked by:

Compiler emitted code for HAL/S construct of the form:

M, where M is an n x n single precision matrix, and I is an integer literal  $\geq 1$ .

Other library modules:

Execution Time (microseconds): (See below).

Input Arguments:

Tiput Arguments.			
Type	Precision	How Passed	Units
matrix(n,n)	SP	$R4 \rightarrow 0^{th}$ element	y a l <mark>⊶</mark> l
integer(n)	SP	R5	<b>-</b>
integer (power)	SP	R6	, - 11 - 11 - 11 - 11 - 11 - 11 - 11 -
matrix(n,n) workareas Output Results:	SP	<b>R7</b>	
Type	Precision	How Passed	Units
Matrix (n,n)	SP	$R2 \rightarrow 0^{th}$ element	en e

Errors Detected:

Error # Cause Fixup

Comments:

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: Let A = original matrix, R = result matrix, T = temporary matrix.

- 1) R = A A
- 2) locate first one bit in exponent, remove it, remember bit position
- 3) go to step 6
- 4) T = R
- 5) R = T T
- 6) Remove exponent bit at current position, increment position.

  If bit was 0 go to step 9.
- 7) T = R
- 8) R = TA
- 9) If any bits left in exponent, go to step 4, otherwise R is complete.

Execution Time (microseconds):

if power = 2: then 
$$15.6n^3 + 15.2n^2 + 5.8n + 43.8$$

if power > 2: same as in MM17DN except

TMULT = 
$$10.0 + 5.8n + 15.2n^2 + 15.6n^3$$

$$TMOVE = 10.2 + 8.6n^2$$

		MV 6DN	<del></del>
H	AL/S-FC LIBRARY ROUTI	NE DESCRIPTION	
Source Member Name:	MV6DN Size	of Code Area:	24_ Hw
Stack Requirement:	O Hw Data	CSECT Size:	Hw
X Intrin	sic C	Procedure	
Other Library Modul	es Referenced: NONE		
ENTRY POINT DESCRIP	TIONS		
Primary Entry Name:	MV6DN		
Function: Multiplie	es a double prec <b>isio</b> n	mxn matrix by a length n	double pre
vector.			
Invoked by:  X Compiler emitt	ed code for HAL/S cor	estruct of the form:	
	an mxn matrix by a lo		
M V,			
V is	a length n double pro	ecision vector.	
V is Other Library		ecision vector.	
Other Library	Modules:		
Other Library			
Other Library  Execution Time (mic	Modules:		
Other Library  Execution Time (mic	Modules:	19.3 + 26.0n)  How Passed	<u>Uni</u>
Other Library  Execution Time (mic  Input Arguments:  Type  Matrix (m,n)	Modules: croseconds): 12.0 + m( Precision DP	19.3 + 26.0n)  How Passed	<u>Uni</u>
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m)	Modules: croseconds): 12.0 + m( Precision DP DP DP SP	How Passed  R2 +0th element R3 +0th element R5	<b>Un</b> :
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n)	Modules: croseconds): 12.0 + m( Precision DP DP	19.3 + 26.0n)  How Passed  R2 $\rightarrow$ 0th element R3 $\rightarrow$ 0th element	
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m)	Modules: croseconds): 12.0 + m( Precision DP DP DP SP	How Passed  R2 +0th element R3 +0th element R5 R6  How Passed	
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n) Output Results:	Modules: croseconds): 12.0 + m(  Precision DP DP SP SP SP	How Passed  R2 +0th element R3 +0th element R5 R6	
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n) Output Results:  Type	Modules:  croseconds): 12.0 + m(  Precision  DP  DP  SP  SP  SP  Precision	How Passed  R2 →0th element R3 →0th element R5 R6  How Passed  R1 →0 th element	<u>Un</u>
Other Library  Execution Time (mic  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n) Output Results:  Type Vector (m)	Modules:  croseconds): 12.0 + m(  Precision  DP  DP  SP  SP  SP  Precision	How Passed  R2 + 0th element R3 + 0th element R5 R6  How Passed  R1 + 0th element	<u>Un</u>
Other Library  Execution Time (mice  Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (m) Output Results:  Type Vector (m)  Errors Detected:	Modules:  Proseconds): 12.0 + m(  Precision  DP  DP  SP  SP  SP  Precision  DP	How Passed  R2 + 0th element R3 + 0th element R5 R6  How Passed  R1 + 0th element	<u>Un</u>
Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n) Output Results:  Type Vector (m)  Errors Detected:	Modules:  Proseconds): 12.0 + m(  Precision  DP  DP  SP  SP  SP  Precision  DP	How Passed  R2 + 0th element R3 + 0th element R5 R6  How Passed  R1 + 0th element	Uni Uni
Input Arguments:  Type  Matrix (m,n) Vector (n) Integer (m) Integer (n) Output Results:  Type Vector (m)  Errors Detected: Error #	Modules:  Proseconds): 12.0 + m(  Precision  DP  DP  SP SP SP  Precision  DP  Cau	How Passed  R2 + 0th element R3 + 0th element R5 R6  How Passed  R1 + 0th element	<u>Un</u>

Algorithm: Uses 2 nested loop, outer loop selecting rows of matrix, inner loop summing products of vector elements with current row elements.

		MV6D3	
	L/S-FC LIBRARY ROUTIN	Z DESCRIPTION	
на			Hw
ource Member Name:	MV6D3 Size	of Code Area: 22	<del></del> .;
stack Requirement: _	Оф <sub>W</sub> Data	CSECT Size:	O Hw
	ic 🔲	Procedure	
Other Library Module	s Referenced:		
ENTRY POINT DESCRIPT	PIONS		
Primary Entry Name:	MV6D3		
Function: Multipli	es a 3x3 double precis	sion matrix by a length 3	double
vector.			
Invoked by:			
Committee	ed code for HAL/S cons	struct of the form:	
* - where M is	a double precision 3	k3 matrix,	
V is	s a double precision 5	-vector.	
Other Library	Modules:		
Execution Time (mic	roseconds): 304.4		
Input Arguments: Type	Precision	How Passed	
Matrix (3,3)	D₽	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element	
Vector (3)	DP	R3 → 0 element	
Output Results:		How Passed	
Type	Precision	$R1 \rightarrow 0$ element	
Vector(3)	$\mathbf{DP}^{(i)}$ . The $\mathbf{DP}^{(i)}$	KI V GIGMON	
Errors Detected:		Fix	аur
Errors Detected: Error #	Caus	<u>Fix</u>	up
	Caus	<u>Fix</u>	up
Error #			
Error #			
Error #  Comments: Register	s Unsafe Across Call:	R1,R2,R3,R4,R5,R6,R7,F0	,F1,F2,F
Error #  Comments:  Register	s Unsafe Across Call:		,Fl,F2,F matrix,

			MV6SN	_
		. DOUBLE DECORIO	TON	
		ROUTINE DESCRIPT		** -
Source Member Name	MV6SN	Size of Code Ar	ea: 18	Hw Hw
Stack Requirement:	O Hw	Data CSECT Size	:	O Hw
🛛 Intri	nsic	Procedure		
Other Library Modu	les Referenced:	NONE		
ENTRY POINT DESCRI	PTIONS			
Primary Entry Name	MV6SN			
Function: Multipli	es a single pred	cision mxn matrix	by a length n si	ngle preci
vector.				
Invoked by:  X Compiler emit	tted code for HA	L/S construct of	the form:	
* - M V. where M	is an mxn singl	e precision matrix single precision v	<b>)</b>	
Other Library	is a length n y Modules:	single precision v	ector.	
Execution Time (m	icroseconds): 1	1.2 + m(ll.0 + 18.	4n)	
Input Arguments:				
Type (m,n)	Precision SP		w Passed + 0th element + 0th element	Uni
Vector (n) Integer (m)	SP SP	R3 R5		<u>-</u>
Integer (n) Output Results:	SP	R6		
Type	Precision		w Passed	<u>Uni</u>
Vector (m)	SP	R1	→ 0 <sup>th</sup> element	_
Errors Detected:		Cause	Fixup	
Error #				Transport de La
Comments:		all: Rl,R2,R3,R4,R		E2 E4 E5

Algorithm: Exchange contents of R1 and R3 for more efficient addressing.

Uses 2 nested loops, outer loops selecting rows of matrix,

inner loop summing products of vector elements with current
row elements.

		MV6S3	·
	T /G HO (TENDENT DOLLER)	ATT DESCRIPTION	
HA	L/S-FC LIBRARY ROUT	INE DESCRIPTION	
Source Member Name:	MV6S3 Size	of Code Area:	20 Hw
Stack Requirement: _	0 Hw Data	a CSECT Size:	O Hw
X Intrins	ic	Procedure	
Other Library Module	s Referenced: No	ONE	
ENTRY POINT DESCRIPT	PIONS		
Primary Entry Name:	MV6S3		
Function: Multiplies	s a 3x3 single precis	sion matrix by a length 3	single preci
vector.			
Invoked by:		and the state of t	
Compiler emitte	ed code for HAL/S con	nstruct of the form:	
$\widetilde{M}$ $\widetilde{V}$ where M	4 is a single precis:	ion 3x3 matrix,	
7	/ is a single prec <b>is</b>	ion 3-vector.	
and the state of t			
Other Tibrary M	lodules.		
Other Library M	fodules:		
Other Library M	iodules:		
Other Library M			
	roseconds): 137.6		
Execution Time (micronic Input Arguments: Type	roseconds): 137.6  Precision	How Passed	Units
Execution Time (micron Input Arguments:  Type  Matrix (3,3)	Precision SP	How Passed R2 + 0 th element	Units
Execution Time (micronic Input Arguments: Type	roseconds): 137.6  Precision	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0} \text{th element}$ $\text{R3} \rightarrow 0 \text{ element}$	Units -
Execution Time (micron Input Arguments:  Type  Matrix (3,3)  Vector (3)	Precision SP	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0 \text{ th element}}$ $\text{R3} \rightarrow 0 \text{ element}$	Units
Execution Time (micronic Input Arguments:  Type  Matrix (3,3)	Precision SP	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element	
Execution Time (micron Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:	Precision SP SP	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element	
Execution Time (microstructure) Input Arguments:	Precision  SP  SP  Precision	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element	Units - - Units
Execution Time (microson Time)  Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)  Errors Detected:	Precision SP SP Precision SP SP	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element How Passed $R1 \rightarrow 0$ element	Units
Execution Time (micron Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)	Precision  SP  SP  Precision	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element How Passed $R1 \rightarrow 0$ element	
Execution Time (microson Time)  Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)  Errors Detected:	Precision SP SP Precision SP SP	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element How Passed $R1 \rightarrow 0$ element	Units
Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)  Errors Detected:  Error #	Precision SP SP Precision SP Cau	$R2 \rightarrow 0$ th element $R3 \rightarrow 0$ element How Passed $R1 \rightarrow 0$ element	Units
Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)  Errors Detected:  Error #  Comments:  Registers Unsafe  Algorithm: Loops	Precision SP SP Precision SP Cau Across Call: R1,R2,R	R2  o 0 th element $R3  o 0$ th element element  How Passed $R1  o 0$ element  se Fig. $R3, R4, R5, R6, F0, F1, F2, F3$ .	Units  Kup  elements with
Input Arguments:  Type  Matrix (3,3)  Vector (3)  Output Results:  Type  Vector(3)  Errors Detected:  Error #  Comments:  Registers Unsafe  Algorithm: Loops	Precision  SP  SP  Precision  SP  Cau  Across Call: R1,R2,R  3 times, each pass strow elements and s	R2  o 0 th element R3  o 0 element How Passed R1  o 0 element se Fig.	Unit

			VM6	<u>DN</u>
HA	L/S-FC LIBRARY	ROUTINE DESCRI	PTION	
Source Member Name:	VM6DN	Size of Code	Area:	26 Hw
Stack Requirement: _	O Hw	Data CSECT Si	ze:	0 Hw
X Intrins	Lc	Procedu	ire	
Other Library Modules	s Referenced:	NONE	<del>- 1 - 200 - 20 - 20 - 20 - 20 - 20 - 20 </del>	
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	VM6DN			
Function: Multipliematrix.	s length n dou	ble precision v	ector and nxm	double precisio
Invoked by:  X Compiler emitted	d code for HAL	/S construct of	the form:	
V M where V is a	a double preci		n ≠ 3.	
Other Library Mo			, ,	
Execution Time (micro	oseconds): 23.	2 + m(23.2 + 27	.6n)	
Input Arguments:				
Type	Precision		ow Passed	Units
Vector(n) Matrix(n,m)	DP	R2	→oth element →oth element	_
Integer (m)	DP SP	K3	→0° element	
Integer (n) Output Results:	SP		R5	
Type	Precision	<u>H</u>	ow Passed	Units
Vector(m)	DP	R1 -	oth element	
Errors Detected:				
Error #		Cause	<u>Fi</u>	хир
Comments:				
Registers Unsafe A	cross Call: Rl	,R2,R3,R4,R5,R6	,R7,F0,F1,F2,F	3,F4,F5.
Algorithm: Uses 2 ms				
USES 2 HE	sted loops: .oop selects ma	atrix dolumn		
		actix column. acts of vector of	elements with m	atriv alaumn
element		-235 OF AGGEOT	remettes wiftl 1	ECTIX CIOUMI
is attached to the property				

			VM6D3	
	HAL/S-FC LIBRAR	Y ROUTINE DESCRIPTI	ON	: : : :
	Source Member Name: VM6D3	Size of Code Are	a:24	Hw
	Stack Requirement: 0_Hw	Data CSECT Size:		O Hw
	X Intrinsic	☐ Procedure	•	
	Other Library Modules Referenced:	NONE		· ·
<del></del>	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: VM6D3			
	Function: Multiplies a length 3 precision matrix.	double precision v	rector by a 3x3	double
	Invoked by:  X Compiler emitted code for HA	L/S construct of th	e form;	
	V M, where V is a double preci M is a double preci Other Library Modules:			
	Execution Time (microseconds):  Input Arguments:  Type Vector(3)  Precision DP	How R2→	Passed 0 <sup>th</sup> element	Units -
	Matrix (3,3) DP	R3 →	O <sup>th</sup> element	
	Output Results:  Type Precision  Vector (3) DP	The second secon	Passed To <sup>th</sup> element	<u>Units</u>
	Errors Detected: Error #	Cause	Fixup	
	Comments: Registers Unsafe Across Call:	Rl,R2,R3,R4,R5,F0,	F1,F2,F3,F4,F5.	
	Switches R2 to p	vector and matrix here for vector into F0,F2 point to matrix;  E column elements with the column	oy appropriate 2,F4;	loading.

		_	VM6S <b>N</b>	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Na	ame: VM6SN	Size of Code Area:	22	Hw
Stack Requiremen	nt; O Hw	Data CSECT Size:	0	Hw
X Int	crinsic	<pre>Procedure</pre>		
Other Library Mc	odules Referenced:	NONE		
ENTRY POINT DESC	CRIPTIONS			
Primary Entry Na	we: VM6SN	·		
	iply a length n sin ision matrix	gle precision vector	by a nxm single	
$\overline{V}_{M}^{*}$ , where $V_{M}$		/S construct of the sion n-vector, n ≠ 3		
Ocher hibra	ary Modules:			
Execution Time	(microseconds): 1	2.4 + m(19.2 + 18.2n)		
		2.4 + m(19.2 + 18.2n)		
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n)		How Pa		<u>Un</u>
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n) Output Results:	:  Precision SP SP SP SP SP SP SP	How Pa $ \begin{array}{r} R2 \rightarrow 0T \\ R3 \rightarrow 0T \\ R6 \\ R5 \end{array} $	ssed h element h element	
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n) Output Results:  Type	Precision  SP SP SP SP SP SP SP Precision	How Pa R2 → OT R3 → OT R6 R5	ssed h element h element	
Input Arguments:  Type Vector(n) Matrix(n, m) Integer (m) Integer (n) Output Results:  Type Vector(m)	Precision SP	How Pa R2 → OT R3 → OT R6 R5	ssed h element h element ssed	
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n) Output Results:  Type	Precision SP	How Pa R2 → OT R3 → OT R6 R5	ssed h element h element ssed	
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n) Output Results:  Type Vector(m)  Errors Detected Error #	Precision SP	How Pa $ \begin{array}{r} R2 \rightarrow 0t \\ R3 \rightarrow 0t \\ R6 \\ R5 \end{array} $ How Pa $ \begin{array}{r} R1 \rightarrow 0^{th} \end{array} $	ssed h element h element ssed element	<u>Un</u>
Input Arguments:  Type Vector(n) Matrix(n,m) Integer (m) Integer (n) Output Results:  Type Vector(m) Errors Detected Error # Comments:	Precision  SP SP SP SP SP SP SP  Precision SP :	How Pa $ \begin{array}{r} R2 \rightarrow 0t \\ R3 \rightarrow 0t \\ R6 \\ R5 \end{array} $ How Pa $ \begin{array}{r} R1 \rightarrow 0^{th} \end{array} $	ssed h element h element ssed element	Un

and matrix column elements.

REPRODUCIBILITY OF THE ORIGINAL PACES

			VM6S3	_
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTI	ION	
				Hw
Source Member N	ame: VM6S3	Size of Code Are		
Stack Requireme	nt: 0 Hw	Data CSECT Size:		O Hw
🔀 In	trinsic	Procedure		
Other Library M	odules Referenced:	NONE		<u> </u>
ENTRY POINT DES	CRIPTIONS			
Primary Entry N	ame: VM6S3	<u></u>		
Function: Multipercoin	plies a length 3 sin sion matrix.	ngle precision vec	tor by a 3x3 sir	gle
Invoked by:	mitted code for HAI	J/S construct of the	ne form:	
М	is a single precision is a single precision ary Modules:			
Execution Time	(microseconds): 14	11.2		
<b>-</b>				
Input Arguments Type	Precision	How	Passed	Units
Vector(3) Matrix(3,3)	SP SP	R2 → R3 →	0 <sup>th</sup> element 0 <sup>th</sup> element	_
Output Results:				
Type	Precision		Passed th	Units
Vector(3)	SP	Rl→	O <sup>th</sup> element	
Errors Detected	•			
Error #		Cause	Fixup	
Comments:	Unsafe Across Call	• p1 p2 p3 p4 p5 F	በ F1 F2 F3	
kegisters	unsale Across Call	Tichienichianith	O,,	
Algorithm:	addressing new cand summing, by e	looping three time column of matrix for elements of vector contain both input	or explicit mult and storing int	o result.
		s. Then circular s		

appropriate pointer into high HW for use as base.

<del></del>				VO6DN	Ì
. ".					
	IAH	J/S-FC LIBRARY	ROUTINE DESCRIPT	rion	
	Source Member Name: _	VO6DN	Size of Code A	rea:	20 Hw
	Stack Requirement:	0 Hw	Data CSECT Size	e:	O Hw
	▼ Intrinsi	Lc	Procedu	re	
	Other Library Modules	Referenced:	None.		
	ENTRY POINT DESCRIPT	CONS			
	Primary Entry Name:	70	6DN		
	Function: Performs v vectors.	ector outer pr	oduct of two dou	ble precision	
	Invoked by:  X Compiler emitted V1 V2, where V		/S construct of louble precision		4.3.
	Other Library Mo		8 ± n(5 8 ± 24 4		
	Execution Time (mid)	oseconds). Iz	.0 1 11(3.0 1 24.9		
	Input Arguments:				maita
	Type vector(n) vector(m) integer(n) integer(m)	Precision DP DP SP SP	R2 =	w Passed Oth element Oth element R5 R6	Units
	Output Results: Type	Precision	Но	w Passed	Units
	Matrix(n,m)	DP	Rl	→ O <sup>th</sup> element	
	Errors Detected:				
	Error #		Cause	Fi:	xup
	Comments: Registers Unsafe	Across Call: R	1,R2,R3,R4,R5,R6	,R7,F0,F1,F4,F	<b>5.</b>
	Inn V2 Out	er loop multip creating a row er loop moves	te of matrix(n); plies element of w of result matri to next element row of result mat	x. of Vl and move	

			V06D3	-
наі	L/S-FC LIBRARY	ROUTINE DESCRIPT	PION	
Source Member Name:	V06D3	Size of Code Ar	rea: 22	Hw
Stack Requirement:	O Hw	Data CSECT Size	<b>0</b>	Hw
	ic	Procedur	'e	
Other Library Module	s Referenced:	None.		
ENTRY POINT DESCRIPT	IONS			
n i Bakus Namas	VO	6D3		
Primary Entry Name: Function: Computes	restor suter n	roduct of length	3 double precisi	on vector
Function: Computes	AGCCOT Offer 5			
Invoked by:	T T T T TINT	/c construct of	the form:	
X Compiler emitte	d code for HAL	As constitued of	CITC LOLLING	7
		· ·		
V1 V2. where V	l and V2 are d	ouble precision	3-vectors.	
V1 V2, where V	'l and V2 are d	ouble precision	3-vectors.	
	'l and V2 are d	ouble precision	3-vectors.	
V1 V2, where V	'l and V2 are d	ouble precision :	3-vectors.	
	'l and V2 are d	ouble precision	3-vectors.	
Other Library M	l and V2 are d	ouble precision	3-vectors.	
	l and V2 are d	ouble precision :	3-vectors.	
Other Library M	l and V2 are d	ouble precision	3-vectors.	
Other Library M	l and V2 are d	ouble precision	3-vectors.	Uni
Other Library M  Execution Time (micr	l and V2 are d	ouble precision (251.0)	ow Passed	<u>Uni</u>
Other Library M  Execution Time (micr  Input Arguments:  Type	nd V2 are dodules: coseconds): <pre>Precision</pre>	ouble precision (251.0)  Ho	ow Passed + Oth element	<u>Uni</u>
Other Library M  Execution Time (micr  Input Arguments:  Type Vector(3)	nd V2 are doubles: coseconds): <pre>Precision</pre> <pre>DP</pre>	ouble precision (251.0)  Ho	ow Passed	<u>Uni</u>
Other Library M  Execution Time (micr  Input Arguments:  Type	nd V2 are dodules: coseconds): <pre>Precision</pre>	ouble precision (251.0)  Ho	ow Passed + Oth element	<u>Uni</u>
Other Library M  Execution Time (micro  Input Arguments:  Type  Vector(3)  Vector(3)	nd V2 are doubles: coseconds): <pre>Precision</pre> <pre>DP</pre>	ouble precision (251.0)  Ho	ow Passed + Oth element	
Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3) Vector(3)  Output Results:	nodules: coseconds):  Precision DP DP	ouble precision (251.0)  He R2 R3	ow Passed + Oth element	
Other Library M  Execution Time (micro  Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  He R2 R3	ow Passed  - Oth element  - Oth element	
Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3) Vector(3)  Output Results:	nodules: coseconds):  Precision DP DP	ouble precision :  251.0  He R2 R3	ow Passed  Oth element  Oth element	Uni Un:
Other Library M  Execution Time (micro Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  He R2 R3	ow Passed  → Oth element  → Oth element  ow Passed  → Oth element	<u>Un</u>
Other Library M  Execution Time (micro Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  He R2 R3	ow Passed  Oth element  Oth element	<u>Un</u>
Other Library M  Execution Time (micro  Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  Ho R2 R3  Ho	ow Passed  → Oth element  → Oth element  ow Passed  → Oth element	<u>Un</u>
Other Library M  Execution Time (micro  Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  Ho R2 R3  Ho	ow Passed  → Oth element  → Oth element  ow Passed  → Oth element	<u>Un</u>
Other Library M  Execution Time (micro  Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  Ho R2 R3  Ho	ow Passed  → Oth element  → Oth element  ow Passed  → Oth element	<u>Un</u>
Other Library M  Execution Time (micro Input Arguments:	nodules: coseconds):  Precision DP DP Precision	ouble precision :  251.0  Ho R2 R3  Ho	ow Passed  → Oth element  → Oth element  ow Passed  → Oth element	<u>Un:</u>
Other Library M  Execution Time (micro Input Arguments:	nodules:  roseconds):  Precision DP DP DP Precision DP	ouble precision :  251.0  Ho R2 R3  Ho R1	ow Passed  + Oth element  + Oth element  ow Passed  - Oth element  Fixur	<u>Un:</u>
Other Library M  Execution Time (micro Input Arguments:	nodules:  roseconds):  Precision DP DP DP Precision DP	ouble precision :  251.0  Ho R2 R3  Ho R1	ow Passed  + Oth element  + Oth element  ow Passed  - Oth element  Fixur	<u>Un</u>
Other Library M  Execution Time (micro Input Arguments:	I and V2 are diodules:  Coseconds):  Precision DP DP Precision DP Across Call:	ouble precision  251.0  Ho R2 R3  Ho R1  Cause	Passed  Ow Passed  Oth element  Ow Passed  Oth element  Fixup  16,F0,F1.	<u>Un</u>
Other Library M  Execution Time (micro Input Arguments:	I and V2 are diodules:  Coseconds):  Precision DP DP Precision DP Across Call:	ouble precision  251.0  Ho R2 R3  Ho R1  Cause	ow Passed  + Oth element  + Oth element  ow Passed  - Oth element  Fixur	<u>U</u> n

0 Hw Dasic es Referenced: N FIONS  VO6SN  Ates vector outer p precision ve	ize of Code Area:  ata CSECT Size:  Procedure  one.  roduct of 2 single	20	Hw Hw
O Hw Dasic es Referenced: N  FIONS  VO6SN  Ates vector outer p precision vector outer p precision vector outer p precision vector outer p	Procedure  one.  roduct of 2 single ctors,	0	
VO6SN  Ates vector outer precision vector ode for HAL/S of the V1 and V2 are single.	Procedure  one.  roduct of 2 single ctors,		Hw
VO6SN  VO6SN  Ates vector outer p precision ve	roduct of 2 single ctors,		
VO6SN  Ates vector outer p  precision ve  ed code for HAL/S of the VI and V2 are sin	roduct of 2 single ctors,		
VOGSN  Ates vector outer p precision ve  ed code for HAL/S of	ctors,		
precision ve precision ve ed code for HAL/S of Vl and V2 are sin	ctors,		
precision ve precision ve ed code for HAL/S of Vl and V2 are sin	ctors,	. n # 3	
ed code for HAL/S o	construct of the form:	n # 3	
	gle precision n-vectors	= n = / 3	
Modules.		7 11 7 3 4	
	How Passed		<u>Ur</u> Un
SP	$R1 \rightarrow 0$ <sup>th</sup> element		<del></del>
		Diana	
<u>Ca</u>	use	Fixup	
	Precision SP	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccc} \underline{\text{Precision}} & & \underline{\text{How Passed}} \\ & & & \\ \text{SP} & & & \\ \text{RS} & & \\ \underline{\text{RS}} & & \\ \text{RS} & & \\ \underline{\text{Precision}} & & \underline{\text{How Passed}} \\ & & & \\ \text{SP} & & & \\ \text{R1} \rightarrow 0^{\text{th}} \text{ element} \end{array}$

5-183

				VO6	is3	
	HAL/S	G-FC LIBRARY	ROUTINE DESCR	IPTION		
Source M	lember Name:	V06S3	Size of Code	Area:	20	Hw
Stack Re	equirement:	O Hw	Data CSECT S	ize:	0	Hw
Other Li	Intrinsic	Referenced:	Proce	dure		
ENTRY PO	DINT DESCRIPTION	NS .				
Primary	Entry Name:	V06S3				
Function	Calculates	vector oute 3 vecto	r produce of 2 rs.	single pro	ecision ler	ngth
	by: mpiler emitted V2, where V1 a					
Ot	her Library Mod	ules:				
Executi	on Time (micros	econds): 16	50.6			
. ∆.	rguments: ype ector (3) ector (3)	Precision SP . SP		How Passed $R2 \rightarrow 0$ th e	Iement	Units
$oldsymbol{\underline{ au}}$	Results: ype atrix(3,3)	Precision SP		How Passed		<u>Units</u>
	Detected: rror #		<u>Cause</u>		<u>Fixup</u>	
Comment Regis	s: sters Unsafe Aci	coss Call: R	1,R2,R3,R4,R5,	R6,F0,F1.		
Algorit	:hm: Same algo	orithm as VO	6DN except tha	t loop exte	ents	

are set to literally 3.

		VVODN	
	AL/S-FC LIBRARY ROUT	INE DESCRIPTION	
Source Member Name:	VVODN Siz	e of Code Area:	6 Hw
Stack Requirement:	0 Hw Dat	a CSECT Size:	O Hw
	sic	Procedure	
Other Library Module	es Referenced: None		
ENTRY POINT DESCRIP	TIONS		
Primary Entry Name:	VVODN		
Function: Generates a double Invoked by:	precision vector of are the same.	length n, all of whose e	elements
	ed code for HAL/S co	nstruct of the form:	
$\overline{V} = S$ , where $\overline{X}$ Other Library 1	scal	ion vector, S is a doubl ar.	le precision
	AA10D3		
Execution Time (mic	roseconds): 7.0 +	5.1n	
Input Arguments:			
Type	Precision	How Passed	Units
Scalar	DP	<b>F</b> 0	
Integer (n)	SP	<b>R5</b>	
Output Results:			
<u>Type</u>	Precision	How Passed	Units
Vector(n)	DP	$R1 \rightarrow 0^{th}$ element	
Errors Detected:			
Error #	<u>Cau</u>	<u>Fi</u>	xup
Comments:			
Registers	Unsafe Across Call:	R1,R4,R5,F0,F1.	
Algorithm: Uses lo	oop counting down len se element of vector	gth (n); Stores input s on each pass through loc	scalar pp.

		<b>_</b>	VODNP	
engan sa	IAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	THYODAY	NOOTING DESCRIPTION		
Source Member Name:	VVODNP <sub>CP</sub>	Size of Code Area:	6	Hw
Stack Requirement:	O Hw	Data CSECT Size:	0	Hw
ĭ Intrin	nsic	Procedure		
Other Library Modul	es Referenced:	None.		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	VVODNP			
Function: Fills		ouble precision matrix wi	ith a double	
Invoked by:				
	ed code for HAL	/S construct of the form		
$M_{n,*} = 0;$ wh	ere M is a doub	le precision matrix .		
Other Library	Modulos.			
Other Library	Modules:			
Maranakian Mina (mi-				
Execution Time (mic	roseconds): 7.0	+ 7.2n		
	roseconds): 7.0	+ 7.2n		
Input Arguments: Type	Precision	How Passed		<u>Unit</u> s
Input Arguments: Type Scalar	Precision DP	How Passed F0		<u>Units</u>
Input Arguments: Type	Precision DP SP	How Passed		<u>Units</u>
Input Arguments:  Type Scalar Integer(outdel	Precision DP SP	How Passed FO R7		Units
Input Arguments:  Type Scalar Integer(outdel Integer(length	Precision DP SP	How Passed FO R7		<u>Units</u>
Input Arguments:  Type Scalar Integer(outdel Integer(length Output Results:	Precision DP SP SP Precision	How Passed F0 R7 R5		
Input Arguments:  Type Scalar Integer(outdel Integer(length Output Results: Type	Precision DP SP SP Precision	How Passed F0 R7 R5 How Passed		

REPRODUCIBILITY OF THE ORIGINAL PAGE ES POOR

				VVOSN	
		HAL/S-FC LIBRAR	Y ROUTINE DE	SCRIPTION	
Sou	urce Member N	Name: VVOSN	Size of C	ode Area:	6 Hw
Sta	ack Requireme	ent: O Hw	Data CSEC	T Size: 0	Hw
	v II	ntrinsic	☐ Pr	ocedure	
O+1		Modules Referenced:	None		
001	ter prorary r	lodules Referenced.			
ENT	TRY POINT DES	SCRIPTIONS			
Pri	imary Entry D	Name: VVOSN			
Fur	nction: Gener	rates a vector of 1	ength n, all	of whose elements	are the same.
	voked by:	emitted code for HF	AL/S construc	et of the form:	
				ector, S is a single scalar.	e precision
	X Other Lib	cary Modules:			
		VV10s	3		
Exe	ecution Time	(microseconds):	7.0 + 5.6n		
Inj	put Argument				
	Type	Precision		How Passed	Units
	Scalar Integer(n)	SP SP		F0 R5	<del>-</del>
Out	tput Results Type	: Precision		How Passed	Units
	Vector (n)			Rl + 0 <sup>th</sup> elemen	t -
Erı	rors Detected	<b>d:</b>			
	Error #		Cause	<u>Fi</u>	xup
Cot	mments:			발표를 위해 보기를 해보고 되었다. 1980년 - 기계	
		safe Across Call:	R1,R4,R5,F0	,F1.	
	gorithm:	Uses loop counting	down kength	of vector (n); Sto of vector on each p	

		vv	OSNP	
•	S-FC LIBRARY ROUT	THE DESCRIPTION		
				Hw
Source Member Name:				
Stack Requirement:	0 Hw Dat	a CSECT Size:	0	Hw
X Intrinsic	,	Procedure		
Other Library Modules	Referenced: None			
 ENTRY POINT DESCRIPTION	ons			
Primary Entry Name:	VVOSNP			
Function: Moves a s	ingle precision s mn of a single pr	calar to all elements ecision matrix.	s of a	
Invoked by:				
X Compiler emitted	code for HAL/S co	onstruct of the form sion matrix .		
n,*				
Other Library Mo	dules:			
Execution Time (micro	seconds): 7.0 +	6,0n		
Input Arguments:				Units
<u>Type</u> Scalar	Precision SP	How Passed F0		-
Integer (outdel) Integer (length)	SP SP	R7 R5		tern.
Output Results:				*****
Type Vector (length)	Precision SP	$\frac{\text{How Passed}}{\text{Rl} \rightarrow 0^{\text{th}} \text{ el}}$	<b>-</b>	<u>Units</u>
Errors Detected: Error #	<u>Ca</u>	use	Fixup	
Comments: Registers Unsa	fe Across Call:	Rl,R4,R5,R7,F0,F1.		
Each pa	length' times;	tores input scalar in	nto vector	
element	pointed to by Rl	and then increments	Rl by outde	1.

		<u> </u>	<u> </u>	
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	L/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	VVlDN Siz	e of Code Area:	8	Hw
Stack Requirement:	0 Hw Dat	a CSECT Size:	0.	Hw
X Intrins	sic C	Procedure		
Other Library Module	es Referenced:			
ENTRY POINT DESCRIPT				
	TTTT TO DE			
Primary Entry Name:				
Function: Moves a l				
	used to move matric			
Invoked by:	ed code for HAL/S co	mstruct of the form	1	
X = V where	v ic a length-n doub	ole precision vector	,	
A = 17 whole	Y is a length-n dou	ble precision vector	<b>c</b> •	
Other Library	Modules:			
Execution Time (mic	roseconds): 4.2 + 1	0.2n		
	roseconds): 4.2 + 1	0.2n		
Input Arguments: Type	Precision	How Passed		<u>v</u> .
Input Arguments: Type Vector (n)	Precision DP			<u>v</u>
Input Arguments: Type	Precision	How Passed R2 $\rightarrow$ 0th ele		<u>v</u>
Input Arguments:  Type Vector (n) Integer (n)  Output Results:	Precision DP SP	How Passed R2 7 Oth ele R5	ement	
Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type	Precision DP	How Passed R2 $\rightarrow$ 0th ele	ment	
Input Arguments:  Type Vector (n) Integer (n)  Output Results:	Precision DP SP Precision	How Passed R2 → 0th ele R5 How Passed	ment	
Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector (n)  Errors Detected:	Precision DP SP Precision DP	How Passed R2 $\rightarrow$ 0th ele R5  How Passed R1 $\rightarrow$ 0th arg	ment	
Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector (n)	Precision DP SP Precision DP	How Passed R2 → 0th ele R5 How Passed	ment <u>1</u> jument	<u>U</u>
Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector (n)  Errors Detected:	Precision DP SP Precision DP	How Passed R2 $\rightarrow$ 0th ele R5  How Passed R1 $\rightarrow$ 0th arg	ment <u>1</u> jument	
Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type Vector (n)  Errors Detected: Error #	Precision DP SP  Precision DP Ca	How Passed R2 $\rightarrow$ 0th ele R5  How Passed R1 $\rightarrow$ 0th arg	ment <u>1</u> jument	
Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type Vector (n)  Errors Detected: Error #	Precision DP SP Precision DP	How Passed R2 $\rightarrow$ 0th ele R5  How Passed R1 $\rightarrow$ 0th arg	ment <u>1</u> jument	
Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector (n)  Errors Detected: Error #  Comments: Registers I	Precision DP SP  Precision DP Ca	How Passed R2 + Oth ele R5  How Passed R1 + Oth arc	ment jument Fixup	

			/V1D3	<b>.</b>
на	L/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	VV1D3 Si:	ze of Code Area; _	14	Hw
Stack Requirement:	0	a CSECT Size:	0	Hw
X Intrins		Procedure		
Other Library Module				
Other Library Module	S Velefeliced:			
ENTRY POINT DESCRIPT	TONS			
Primary Entry Name:	VV1D3			
Function: Moves a c	double precision 3-	vector.		
Invoked by:			• 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
X Compiler emitte	a code for HAT./S c	onstruct of the fo	rm:	
V COMPTTET SWITCH	d code for ind/o	0110 01 000 00		
	X and Y are double			
$\overline{X} = \overline{Y}$ , where $X$	X and Y are double			
	X and Y are double			
$\overline{X} = \overline{Y}$ , where $X$	X and Y are double			
$\overline{X} = \overline{Y}$ , where $X$	X and Y are double			
X = Y, where X  Other Library M  Execution Time (micro	X and Y are double			
$\overline{X} = \overline{Y}$ , where $X$	X and Y are double	precision 3-vector	s	
X = Y, where Y  Other Library M  Execution Time (micro  Input Arguments:	X and Y are double Modules: roseconds): 25.2	precision 3-vector	s	
X = Y, where Y  Other Library M  Execution Time (micro  Input Arguments:  Type	X and Y are double Modules: roseconds): 25.2 Precision	precision 3-vector	s	Un
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:	X and Y are double Modules: roseconds): 25.2  Precision DP	precision 3-vector $\frac{\text{How Pass}}{\text{R2} \rightarrow \text{0th el}}$	sed ement	
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type  Vector(3)	X and Y are double Modules: roseconds): 25.2 Precision	precision 3-vector	sed ement	
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:  Type Vector(3)	X and Y are double  Modules:  roseconds): 25.2  Precision  DP  Precision	precision 3-vector $\frac{\text{How Pass}}{\text{R2} \to \text{0th el}}$	sed ement	
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:  Type Vector(3)  Errors Detected:	X and Y are double  Modules:  roseconds): 25.2  Precision DP  Precision DP	precision 3-vector $\frac{\text{How Pass}}{\text{R2} \to \text{0th el}}$	sed ement	
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:  Type Vector(3)	X and Y are double  Modules:  roseconds): 25.2  Precision DP  Precision DP	Precision 3-vector  How Pass  R2 $\rightarrow$ 0th el  How Pass  R1 $\rightarrow$ 0th el	sed ement sed ement	
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:  Type Vector(3)  Errors Detected:	X and Y are double  Modules:  roseconds): 25.2  Precision DP  Precision DP	Precision 3-vector  How Pass  R2 $\rightarrow$ 0th el  How Pass  R1 $\rightarrow$ 0th el	sed ement sed ement	<u>Un</u>
X = Y, where X  Other Library M  Execution Time (micro  Input Arguments:  Type Vector(3)  Output Results:  Type Vector(3)  Errors Detected:	X and Y are double  Modules:  roseconds): 25.2  Precision DP  Precision DP	Precision 3-vector  How Pass  R2 $\rightarrow$ 0th el  How Pass  R1 $\rightarrow$ 0th el	sed ement sed ement	

REPRODUCIBILITY OF THE ORIGINAL PAGE TO BE A SECOND OF THE

				<del></del>
	HA	L/S-FC LIBRARY RO	UTINE DESCRIPTION	
Source Memb	er Name:	VV1D3P s	ize of Code Area: 18	Hw
	7		ata CSECT Size:	0 Hw
			ata CSECT SIZE:	, <b>nw</b>
	Intrins:	lc .	☐ Procedure	
Other Libra	ry Module:	Referenced:	None	
ENTRY POINT	DESCRIPT:	IONS		
Primary Ent	ry Name:	VV1D3P		
		ength 3 double pre cor or row or colu	ecision vector or row or commof a matrix.	lumn of a
Invoked by:				
X Compil			construct of the form:	
X Compil			construct of the form:  3 matrix and V is a 3-vecto	<b>r.</b>
<u>v</u> =	M <sub>1,*</sub>	where M is a 3x3		<b>r.</b>
Compil V =		where M is a 3x3		
$\overline{V}$ Compil $\overline{V}$ =	M 1,* Library M	where M is a 3x3		
$\overline{V}$ Compil $\overline{V}$ =	M 1,* Library M	where M is a 3x3 odules: oseconds): 46.0 i	matrix and V is a 3-vecto	is contiguous
$\overline{\overline{V}}$ Compil $\overline{\overline{V}}$ = Other Execution T	M <sub>1,*</sub> Library Mo	where M is a 3x3 odules: oseconds): 46.0 i . 48.4 i	matrix and V is a 3-vecto f neither input nor output f either input or output i	is contiguous s contiguous.
$\overline{\overline{V}}$ Compil $\overline{\overline{V}}$ = Other Execution T Input Argum	Ml,* Library Mo ime (micro	where M is a 3x3 odules: oseconds): 46.0 i	matrix and V is a 3-vector f neither input nor output f either input or output in the How Passed	is contiguous
Compil  V =  Other  Execution T  Input Argum  Type Vector(3)	M <sub>1,*</sub> Library Mo ime (micro	where M is a 3x3 odules: oseconds): 46.0 i . 48.4 i	matrix and V is a 3-vecto f neither input nor output f either input or output i	is contiguous s contiguous.
$\overline{\overline{V}}$ Compil $\overline{\overline{V}}$ = Other Execution T Input Argum	I,* Library Mo ime (micro ents: (Indel)	where M is a 3x3 odules: oseconds): 46.0 i	f neither input nor output f either input or output in the How Passed R2 - Oth elament	is contiguous s contiguous.
Compil  V =  Other  Execution T  Input Argum  Type  Vector(3)  Integer  Integer	Indel)	where M is a 3x3 odules:  oseconds): 46.0 i	f neither input nor output f either input or output in the How Passed R2 - Oth element R6	is contiguous s contiguous.
Compil  V =  Other  Execution T  Input Argum  Type  Vector(3)  Integer  Integer	Indel) (Outdel)	where M is a 3x3 odules:  oseconds): 46.0 i	f neither input nor output f either input or output in the How Passed R2 - Oth element R6	is contiguous s contiguous. <u>Units</u>
Compil  V =  Other  Execution T  Input Argum  Type  Vector(3) Integer  Integer  Cutput Resu	Indel) (Outdel)	where M is a 3x3 odules:  oseconds): 46.0 i	f neither input nor output f either input or output in the How Passed R2 + Oth elament R6 R7	is contiguous s contiguous. <u>Units</u>
Compil  V =  Other  Execution T  Input Argum  Type  Vector(3) Integer Integer Coutput Resu  Type	Indel) (Outdel)	where M is a 3x3 odules:  oseconds): 46.0 i	f neither input nor output f either input or output in the How Passed R2 + Oth element R6 R7	is contiguous s contiguous.

1717	D3P	
$\Lambda \Lambda T$	שניע.	

Secondary Entry Name:	AATDNA		
Function: Moves lend a matrix to a row	gth n double precision or column vector.	vector or row or column	of
Invoked by:			
$\overline{V} = \overline{M}_{1,*}$ , where	code for HAL/S constr M is an nxm matrix, V i	uct of the form: is an n-vector.	
Other library mo	dules:		
Execution Time (micro contigu	seconds): 11.4n + 10.2 nous. 11.4 + 12.6 if e	if neither input nor ou ither input or output is	tput is contiguous.
Input Arguments:			
Type	Precision	How Passed R2 → 0 <sup>th</sup> element	<u>Units</u>
Vector (n) Integer (Indel) Integer (Outdel) Integer (n)	DP SP SP SP	R2	
Output Results:		How Passed	Units
Type	Precision	$R1 \rightarrow 0^{th}$ element	Unites
Vector (n)	DP	KT -> 0 Elemente	
Errors Detected:			
· Error #	Cause	Fixup	<u>.</u>
Commonts.		en de de la filipi el militar de la figura (general el que	

Registers Unsafe Across Call: R1,R2,R4,R5,R6,R7,F0,F1.

Algorithm: Tests outdel, of 0, sets it to 4(HW); Tests indel, if 0, sets it to 4(HW); Loops 'length' times, adding indel to input pointer and outdel to output pointer each time. Each loop moves current input element to current output element.

		W	/lsn	
H	AL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	VVlsNSiz	e of Code Area:	8	_ Hw
Stack Requirement:	O Hw Dat	a CSECT Size:	0	Hw
X Intrin		Procedure		-
Other Library Modul	es Referenced: Nor	16		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	VVlsn			
	length-n single preci	sion westor		
	Also used to move ma			
Other Library		ngle precision vectors		
Other Library				
Other Library  Execution Time (mic	Modules:			
Other Library  Execution Time (mic  Input Arguments:  Type	Modules: roseconds): 4.2 + 7  Precision	.8n <u>How Passed</u>		
Other Library  Execution Time (mic  Input Arguments:  Type  Vector (n)	Modules: roseconds): 4.2 + 7	.8n		<u>Un</u>
Other Library  Execution Time (mic  Input Arguments:  Type  Vector (n)  Integer (n)	Modules: roseconds): 4.2 + 7  Precision SP	.8n $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ elemen}}$		
Other Library  Execution Time (mic  Input Arguments:  Type  Vector (n)	Modules: roseconds): 4.2 + 7  Precision SP	.8n $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ elemen}}$		<u>Un</u>
Other Library  Execution Time (mic  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:	Modules: roseconds): 4.2 + 7  Precision SP SP	.8n  How Passed  R2 $\rightarrow$ 0th elemen  R5	<b>1</b>	
Other Library  Execution Time (mice  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:  Type	Modules: roseconds): 4.2 + 7  Precision SP SP SP Precision	.8n $\frac{\text{How Passed}}{\text{R2} \to 0^{\text{th}} \text{ elemen}}$ R5	<b>1</b>	<u>Un</u>
Other Library  Execution Time (mice  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:  Type  Vector (n)	Modules: roseconds): 4.2 + 7  Precision SP SP SP Precision	.8n  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> elemen  R5  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> elemen	<b>1</b>	<u>U</u> m
Other Library  Execution Time (mice Input Arguments:	Modules: roseconds): 4.2 + 7  Precision SP SP SP Precision SP	.8n  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> elemen  R5  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> elemen	a <b>t</b>	<u>U</u> m
Other Library  Execution Time (mic  Input Arguments:	Modules: roseconds): 4.2 + 7  Precision SP SP SP Precision SP	.8n  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> elemen  R5  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> elemen	a <b>t</b>	<u>Ur</u>

		· <u></u>	VV1S3	
	/S-FC LIBRARY ROUTI			
Source Member Name: _	VV1S3 Size	of Code Area:	8	— Hw
Stack Requirement:	0 Hw Data	CSECT Size:	0.	Hw
Intrinsi	Lc C	Procedure		
Other Library Modules	Referenced: No	ie.		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	VV1S3			
Function: Vector mov	ve, length 3, single	precision.		
Invoked by:				
Compiler emitte	ed code for HAL/S co	nstruct of the I	· imtc	•
-1 COMPTTOT OFF			-4-20	
Ti - V2 where	V1 and V2 are sing.	Le precision 3-ve	Crora.	
$\overline{V1} = \overline{V2}$ , where	V1 and V2 are sing.	Le precision 3-ve	GCOT2.	
$\overline{V1} = \overline{V2}$ , where	V1 and V2 are sing.	Le precision 3-ve	C. C	
$\overline{V1} = \overline{V2}$ , where Other Library N	V1 and V2 are sing.	Le precision 3-ve	CCOIS	
$\overline{V1} = \overline{V2}$ , where $\Box$ Other Library M	V1 and V2 are sing.	e precision 3-ve		
$\overline{V1} = \overline{V2}$ , where $\Box$ Other Library M	V1 and V2 are sing.	Le precision 3-ve		
$\overline{V1} = \overline{V2}$ , where	V1 and V2 are sing.	Le precision 3-ve		
$\overline{V1} = \overline{V2}$ , where $\square$ Other Library N	V1 and V2 are sing. Modules: roseconds): 16.8	How Pas	sed	un de la companya de
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type	V1 and V2 are sing. Modules: roseconds): 16.8  Precision	Le precision 3-ve $\frac{\text{How Pas}}{\text{R2} \to \text{Oth}}$	sed	<u>Un</u>
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:	V1 and V2 are sing. Modules: roseconds): 16.8	How Pas	sed	<u>Un</u>
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)	V1 and V2 are sing. Modules: roseconds): 16.8  Precision	$\begin{array}{c} \text{How Pas} \\ \text{R2} \rightarrow \text{0th} \end{array}$	ssed element	
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP	How Pas $R2 \rightarrow 0$ th $R2 \rightarrow 0$	ssed element	
VI = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:  Type	V1 and V2 are sing. Modules: roseconds): 16.8  Precision	$\begin{array}{c} \text{How Pas} \\ \text{R2} \rightarrow \text{0th} \end{array}$	ssed element	
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP	How Pas $R2 \rightarrow 0$ th $R2 \rightarrow 0$	ssed element	
V1 = V2, where  Other Library M  Execution Time (mic: Input Arguments: Type Vector (3)  Output Results: Type	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP	How Pas $R2 \xrightarrow{\text{How Pas}} \uparrow 0 \text{ th}$ $R1 \xrightarrow{\text{How Pa}} \uparrow 0 \text{ th}$	ssed element	Uı
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:  Type Vector (3)	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP	How Pas $R2 \rightarrow 0$ th $R2 \rightarrow 0$	ssed element ssed element	ַוּטַ
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:  Type Vector (3)  Errors Detected:	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP  Precision	How Pas $R2 \xrightarrow{\text{How Pas}} \uparrow 0 \text{ th}$ $R1 \xrightarrow{\text{How Pa}} \uparrow 0 \text{ th}$	ssed element ssed element	Ur
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:  Type Vector (3)  Errors Detected: Error #	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP  Precision	How Pas $R2 \xrightarrow{\text{How Pas}} \uparrow 0 \text{ th}$ $R1 \xrightarrow{\text{How Pa}} \uparrow 0 \text{ th}$	ssed element ssed element	Ur
V1 = V2, where  Other Library M  Execution Time (mic:  Input Arguments:  Type Vector (3)  Output Results:  Type Vector (3)  Errors Detected:  Error # None	V1 and V2 are sing. Modules: roseconds): 16.8  Precision SP  Precision	How Pas R2 $\rightarrow$ 0th R1 $\rightarrow$ 0th	ssed element element Fixur	<u>Un</u>

REPRODUCIBILITY OF THE URIGINAL PAGE IS FUOR

HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
		Hw
Source Member Name: VV1S3P		¥7- •
Stack Requirement: 0 Hw	Data CSECT Size: 0	Hw
▼ Intrinsic	☐ Procedure	
Other Library Modules Referenced:	None	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: VV1S3P		
Function: Moves an SP 3-vector ( a 3-vector (or row or column of	or row or column of a matrix a matrix) when elements are	) to not contiguous.
V = M , where M is a 3x V is a 3-  Other Library Modules:	3 matrix, and -vector.	
Execution Time (microseconds):	38.4 if neither input nor out 40.8 if either input or outpu	put is contiguou t is contiguous.
Input Arguments:		Units
Type Precision	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$	
Vector(3) SP Integer(indel) SP	R6	Hw
Integer(outdel) SP	R7	Hw
Output Results:	How Passed	Units
Type Precision	+	
Type Precision Vector(3) SP	$Rl \rightarrow 0$ <sup>th</sup> element	
<u>-12</u>		
Vector(3) SP		Fixup
Vector(3) SP  Errors Detected: Error #	<u>Cause</u> of size for use by VVISNP code	Fixup

Errors Detected: Error # None	Ca	wse Fix	up.
Input Arguments:  Type Vector(n) Integer (indel) Integer (outdel) Integer (length) Output Results:  Type Vector(n)	Precision SP SP SP SP SP SP SP	How Passed R2 → 0th element R6 R7 R5 . How Passed R1 → 0th element	Units Hw Hw Units
Other library mode Execution Time (micros contiguous.	2 6 6 4 1	0.2 if neither input nor outher input or output is cont	itput is tiguous.
	ere M is an n x m m	nstruct of the form: matrix, and V is an n-vector	
Invoked by:			
Secondary Entry Name: Function:	h-n single precisi	on,	

Comments:

Registers Unsafe Across Call: Rl,R2,R4,R5,R6,R7,F0,F1.

Tests outdel, if 0, sets it to 2 (halfwords). Tests indel, if 0, sets it to 2 (halfwords), Algorithm:

Loops 'length' times, adding indel to input pointer and outdel to output pointer each time. Each loop moves current input element to current output element.

			LTN
•	HAL/S-FC LIBRARY ROU	TINE DESCRIPTION	
Source Member Name	:	ze of Code Area:	8 Hw
Stack Requirement:	0 Hw Da	ta CSECT Size:	O Hw
✓ Intri	nsic	☐ Procedure	
Other Library Modu	les Referenced:		
ENTRY POINT DESCRI	PTIONS		
Primary Entry Name:	. VVITN		
Function: Moves a precision		cision vector and conv	erts it to sing
Invoked by:	Varao gaed	to move matrices);	
-	ted code for HAL/S co	onstruct of the form:	
$\overline{V}1 = \overline{V}2$ , where		louble precision vecto	
		vector of declared len	gth n.
Other Library	Modules:		
Execution Time (mid	croseconds): 4.2 +	9.0n	
	croseconds): 4.2 +	9.0n	
Input Arguments: Type	croseconds): 4.2 +	How Passed	
Input Arguments:  Type Vector (n)	Precision DP	How Passed R2 → Oth elemen	t'
Input Arguments: Type	Precision	How Passed	
Input Arguments:  Type Vector (n)	Precision DP	How Passed R2 → Oth elemen	
Input Arguments:  Type Vector (n) Integer (n)	Precision DP	How Passed R2 → Oth elemen R5 How Passed	t' <u>Un</u>
Input Arguments:  Type Vector (n) Integer(n)  Output Results:	Precision DP SP	R2 + Oth elemen R5	t' <u>Un</u>
Input Arguments:  Type Vector (n) Integer(n)  Output Results: Type	Precision DP SP Precision	How Passed R2 → Oth elemen R5 How Passed	t' <u>Un</u>
Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type Vector (n)  Errors Detected:	Precision DP SP Precision SP	$\begin{array}{c} \text{How Passed} \\ \text{R2} \rightarrow 0^{\text{th}} \text{ elemen} \\ \text{R5} \\ \\ \hline \text{How Passed} \\ \text{R1} \rightarrow 0^{\text{th}} \text{ elemen} \end{array}$	<b>t</b>
Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type Vector (n)	Precision DP SP Precision SP	How Passed R2 → Oth elemen R5 How Passed	<b>t</b>
Input Arguments:  Type Vector (n) Integer(n)  Output Results: Type Vector (n)  Errors Detected: Error #  Comments:	Precision DP SP Precision SP	How Passed R2 $\rightarrow$ 0 <sup>th</sup> elemen R5  How Passed R1 $\rightarrow$ 0 <sup>th</sup> elemen	t' <u>Un</u> t

		<u> </u>	r3	
на	L/S-FC LIBRARY ROU'	TINE DESCRIPTION		
Source Member Name:	Si	ze of Code Area:	12	- Hw
Stack Requirement: _		ta CSECT Size:	0	_ Hw
X Intrins	sic	☐ Procedure		
Other Library Module	s Referenced:			
ENTRY POINT DESCRIPT	PIONS		The second of th	
Primary Entry Name:	VV1T3-			
Function: Moves a l single pr		ision vector and conve	erts it to	
Invoked by:				
Compiler emitte	ed code for HAL/S C	construct of the form: precision 3-vector, and		
V1 = V2, whe	v2 is a double r	precision 3-vector.		
Other Library				
Execution Time (mic	roseconds): 21.2			
Input Arguments:				Ur
Type	Precision	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ elemen}}$	ŧ	01
Vector (3)	DP			
Output Results:				
Output Results: Type	Precision	How Passed	ise i situ i jihalika <b>¥</b> ekalas	<u>U</u>
	Precision SP	$\frac{\text{How Passed}}{\text{Rl} \to 0^{\text{th}} \text{ elemen}}$	t	<u>U</u> 1
Type Vector (3) Errors Detected:	SP	$Rl \rightarrow 0$ th elemen		<u>U</u> 1
Type Vector (3)	SP		t <u>Fixup</u>	
Type Vector (3) Errors Detected:	SP	$Rl \rightarrow 0$ th elemen		Ut
Type Vector (3)  Errors Detected: Error #  Comments:	SP	$Rl \rightarrow 0$ th elemen	Fixup	

REPRODUCIBILITY OF THE ORIGINAL PAGE IN POOR

HAL	/S-FC LIBRARY ROUTI	NE DESCRIPTION	
Source Member Name:		of Code Area: 14	1
Source Member Name:			
Stack Requirement:	O Hw Data	CSECT Size:	0 1
Intrinsic	c 🗖	Procedure	
Other Library Modules	Referenced: NONI	3	·
ENTRY POINT DESCRIPTION	ons		
Primary Entry Name:	VVlT3P		
to a single	precision vector of	ion vector or row or colum r row or column of a matri	x.
Invoked by:			
x Compiler emitted	code for HAT/S con	struct of the form:	
X Compiler emitted	in a dingle procin	ion 3-vector and M is a do	uhle
V=M Wnara V	TE STUGIE PRECIS	TOUR 3-AGE FOR STIR IN TR S OF OR	ante l
1,* 3x3 mat	rix.		
	rix.		
1,* 3x3 mat	rix.		
Other Library Mo	rix. dules:	neither input nor output i	s cont
Other Library Mo	rix. dules:	neither input nor output is	s cont
1,* 3x3 mat  Other Library Mo  Execution Time (micro	rix. dules:	neither input nor output is either input or output is	s con conti
Other Library Mo  Execution Time (micro  Input Arguments:	rix. dules: seconds): 38.4 if 1 40.8 if	either input or output is	s conti
Other Library Mo  Execution Time (micro  Input Arguments:	rix. dules: seconds): 38.4 if 1 40.8 if 6	How Passed	s con conti
Type Vector(3) Integer (indel)	rix. dules: seconds): 38.4 if 1 40.8 if	How Passed R2 + 0th element R6	s contid
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3)	rix. dules: seconds): 38.4 if 1 40.8 if 6	How Passed R2 + Oth element	s contig
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)	rix. dules: seconds): 38.4 if 1 40.8 if 6	How Passed R2 + Oth element R6 R7	s conti
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type	rix. dules: seconds): 38.4 if 1 40.8 if 6	How Passed R2 → Oth element R6 R7  How Passed	s conti
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer (outdel)  Output Results:	rix. dules: seconds): 38.4 if 1 40.8 if 6	How Passed R2 + Oth element R6 R7	s conti
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type	rix. dules: seconds): 38.4 if a 40.8 if conds  Precision OP SP SP SP Precision	How Passed R2 + Oth element R6 R7  How Passed R1 + Oth element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type Vector(3)	rix. dules: seconds): 38.4 if a 40.8 if conds  Precision OP SP SP SP Precision	How Passed R2 + Oth element R6 R7  How Passed R1 + Oth element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type Vector(3)  Errors Detected:	dules: seconds): 38.4 if a 40.8 if of the seconds of the second of the seconds of the second o	How Passed R2 + Oth element R6 R7  How Passed R1 + Oth element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type Vector(3)  Errors Detected:	dules: seconds): 38.4 if a 40.8 if of the seconds of the second of the seconds of the second o	How Passed R2 + Oth element R6 R7  How Passed R1 + Oth element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector(3) Integer(indel) Integer(outdel)  Output Results:  Type Vector(3)  Errors Detected:	dules: seconds): 38.4 if a 40.8 if of the seconds of the second of the seconds of the second o	How Passed R2 + Oth element R6 R7  How Passed R1 + Oth element	

$\tau \pi \tau$	סגייח	

Secondary Entry Name:	VVITNP		
	le precision length o w or column vector.	r row column vector to a s	ingle precision
Invoked by:			
Z Compiler emitted	code for HAL/S const	ruct of the form:	
V=M <sub>1</sub> ,* Where V is	a declared lenth n suble precision nxm mat	single precision vector and	
Other library mod	and the second s		
The state of the s			
Execution Time (microscontiguous. 8	seconds): 8.6n + 10. 3.6n + 12.6 if either	2 if neither input nor outp input or output is contigu	out is lous.
Execution Time (micros contiguous. 8  Input Arguments:	3.6n + 12.6 if either	input or output is contigu	ious.
contiguous. 8  Input Arguments:  Type	3.6n + 12.6 if either  Precision	How Passed	out is nous. Units
Input Arguments:  Type Vector (n)	3.6n + 12.6 if either	input or output is contigu	ious.
Input Arguments:  Type  Vector (n) Integer (outdel) Integer (indel)	Precision  DP SP SP	How Passed  R2 + 0th element  R7  R6	ious.
Input Arguments:  Type  Vector (n) Integer (outdel) Integer (indel Integer (n)	Precision DP SP	How Passed R2 + 0th element R7	ious.
Input Arguments:  Type  Vector (n) Integer (outdel) Integer (indel)	Precision  DP SP SP	How Passed  R2 + 0th element  R7  R6	ious.
Input Arguments:  Type  Vector (n) Integer (outdel) Integer (indel Integer (n) Output Results:	Precision  DP SP SP SP SP	How Passed  R2 + 0th element  R7  R6  R5	Units
Input Arguments:  Type  Vector (n) Integer (outdel) Integer (indel Integer (n) Output Results: Type	Precision  DP SP SP SP SP SP Precision	How Passed  R2 + 0th element  R7  R6  R5	Units

## Comments:

Registers Unsafe Across Call: R1,R2,R4,R5,R6,R7,F0,F1.

## Algorithm:

If outdel=0, set to 2 (HW);
If indel=0, set to 4(HW);

Loops 'length' times, adding indel to input pointer and outdel to output pointer each time. Each loop moves current

input element to current output element.

			VV1WN.	
на	L/S-FC LIBRARY ROU	TINE DESCRIPTION		
Source Member Name:	VVlWN Si	ze of Code Area	10	Hw
Stack Requirement: _	O Hw Da	ta CSECT Size:	0	Hv
Intrins	ic	Procedure		
Other Library Module	<b>N1</b> .	one.		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:				
Function: Moves a le	ngth-n single prec	ision vector and	converts it t	o doul
р	recision. (Also	used to move mat	rices).	
Invoked by:				
X Compiler emitte	ed code for HAL/S	construct of the	ioim:	
$\overline{X} = \overline{Y}$ , where Y	is a single preci	sion length n ve	cuor,	
v	, ta a double preci	sion length n ve	ctor.	
X	is a double preci	sion length n ve	ctor,	
Other Library N	is a double preci	sion length n ve	ctor,	
Other Library N	( is a double preci	SION TENGEN IN VO	ctor,	
X	( is a double preci	SION TENGEN IN VO	ector,	
Other Library M  Execution Time (michae)  Input Arguments:	( is a double preci	9.0n		
Other Library M  Execution Time (michael Michael Micha	(is a double preci- Modules:  roseconds): 8.4 +  Precision	9.0n	?assed	
Other Library M  Execution Time (mich	( is a double preci	9.0n How I	?assed	
Other Library M  Execution Time (mich  Input Arguments:  Type  Vector (n)  Integer (n)	( is a double preci- Modules:  roseconds): 8.4 +  Precision  SP	9.0n $\frac{\text{How I}}{\text{R2} \to 0^{\frac{1}{2}}}$	Passed element	
Other Library M  Execution Time (mich  Input Arguments:  Type  Vector (n)	( is a double preci- Modules:  roseconds): 8.4 +  Precision  SP	9.0n $\frac{\text{How I}}{\text{R2}} \rightarrow \frac{\text{Oth}}{\text{R5}}$ How	Passed Passed	
Other Library M  Execution Time (mich  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:	(is a double preci- Modules:  roseconds): 8.4 +  Precision  SP  SP	9.0n $\frac{\text{How I}}{\text{R2} \to 0^{\frac{1}{2}}}$	Passed Passed	
Other Library M  Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type	Modules:  roseconds): 8.4 +  Precision  SP  SP  Precision  DP	9.0n  How I  R2 $\rightarrow$ 0th  R5  How $\rightarrow$ 0th	Passed Passed element	
Other Library M  Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector(n)	Modules:  roseconds): 8.4 +  Precision  SP  SP  Precision  DP	9.0n $\frac{\text{How I}}{\text{R2}} \rightarrow \frac{\text{Oth}}{\text{R5}}$ How	Passed Passed	
Other Library M  Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector(n)  Errors Detected:	Modules:  roseconds): 8.4 +  Precision  SP  SP  Precision  DP	9.0n  How I  R2 $\rightarrow$ 0th  R5  How $\rightarrow$ 0th	Passed Passed element	
Other Library M  Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector(n)  Errors Detected: Error #	Modules:  roseconds): 8.4 +  Precision  SP  SP  Precision  DP	9.0n  How I  R2 $\rightarrow$ 0th  R5  How $\rightarrow$ 0th	Passed Passed element	
Other Library M  Execution Time (micro  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:  Type  Vector(n)  Errors Detected:  Error #  Comments:	Modules:  roseconds): 8.4 +  Precision  SP  SP  Precision  DP	9.0n  How I  R2 $\rightarrow$ 0th  R5  How :  R1 $\rightarrow$ 0th	Passed Passed element	

			VV1W3	
НA	L/S-FC LIBRARY	ROUTINE DESCRIP	TION	
Source Member Name:	VV1W3	Size of Code A	rea: <u>. 12</u>	Hw
Stack Requirement: _	O Hw	Data CSECT Siz	e:	) Hw
☐ Intrins	ic	☐ Procedu	re	
Other Library Module	s Referenced:	None.		
ENTRY POINT DESCRIPT Primary Entry Name:	VV1W3			
Function: Moves a l	ength 3 single		r and converts	s it to
	Y is a single X is a double Modules:	/S construct of precision 3-vect precision 3-vect	or, and	
Input Arguments: Type	Precision	<u> H</u> c	ow Passed	<u>Units</u>
Vector(3)	SP	R2	→ 0 <sup>th</sup> element	
Output Results:				
Type	Precision		w Passed	Units
	. DP	R1	. → 0 <sup>ch</sup> elemen	<b>t</b> aring for the second
Errors Detected: Error #		Cause	<u>Fi</u>	xup
Vector(3) Errors Detected: Error # Comments:	DP		→ 0 <sup>th</sup> element	
Registers Un	nsafe Across Ca	11: R1,R2,R4,F0,	F1.	
The		e to load (SP) ea (DP) into each e		

HAL Source Member Name: Stack Requirement:  X Intrinsi Other Library Modules		Size of (	Code Area:	18
Stack Requirement:	O Hw			18
X Intrinsi		Data CSE	CT Size:	
🛚 Intrinsi				0
	C		cocedure	
Other Library Modules		П г	Ocedure	
	Referenced:	None		
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name:	VV1W3P			
				- column of
Function: Moves a sir matrix, to a double p	gle precisio recision vec	n length 3 t tor <b>or</b> row (	rector or row of or column of a n	natrix.
Invoked by:  X Compiler emitted	code for HA	I/S constru	ct of the form:	
	V is a doub			
$V = M_{1,*}$ , where	v is a doub precision 3	le precision	n 3-vector, M is	
	precision 3	le precision		
Other Library Mc	precision 3	le precision		
Other Library Mo	precision 3	le precision	n 3-vector, M is	s a single
	precision 3 dules:	le precision x3 matrix.  44.8 if neit	n 3-vector, M is	s a single output is co
Other Library Mo	precision 3 dules:	le precision x3 matrix.  44.8 if neit	n 3-vector, M is	s a single output is co
Other Library Mo	precision 3 dules:	le precision x3 matrix.  44.8 if neit	ther input nor oner input or out	s a single output is co cput is cont
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (3)	precision 3 dules: pseconds):  Precision SP	le precision x3 matrix.  44.8 if neit	ther input nor oner input or out  How Passed  R2 $\rightarrow$ 0th element	s a single output is co cput is cont
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (3) Integer (indel)	precision 3 dules:  pseconds):  Precision SP SP	le precision x3 matrix.  44.8 if neit	ther input nor oner input or out  How Passed  R2 → Oth elements	s a single output is co cput is cont
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (3) Integer (indel) Integer (outdel)	precision 3 dules: pseconds):  Precision SP	le precision x3 matrix.  44.8 if neit	ther input nor oner input or out  How Passed  R2 $\rightarrow$ 0th element	s a single output is co cput is cont
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (3) Integer (indel) Integer (outdel)  Output Results:	precision 3 dules: seconds):  Precision SP SP SP	le precision x3 matrix.  44.8 if neit	ther input nor oner input or out  How Passed  R2 → Oth elements	s a single output is co cput is cont
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (3) Integer (indel) Integer (outdel)	precision 3 dules:  pseconds):  Precision SP SP	le precision x3 matrix.  44.8 if neit	ther input nor other input or out the respective of the respective	s a single output is co put is cont
Other Library Mo  Execution Time (micro  Input Arguments:      Type     Vector (3)     Integer (indel)     Integer (outdel)  Output Results:     Type	precision 3 dules:  precision SP SP SP SP Precision	le precision x3 matrix.  44.8 if neit	ther input nor other input or out the in	s a single output is co put is cont

VV1W3P		

Secondary Entry Name:	VV1WNP		
Function: Moves a ler double precis	ngth n s <b>in</b> gle ion column vec	precision row or column vector to a tor.	
Invoked by:			
<pre>Compiler emitted of v = M<sub>1,*</sub>, where</pre>	v is declared	S construct of the form: I as a length n double precision vector single precision matrix.	, and
Other library mod	ules:		
Execution Time (micros contiguous. 10	econds): 10.2r	n + 15.0 if either input or output is neither input nor output is contiguous	<b>3.</b>
Input Arguments:			
Туре	Precision	How Passed R2 → Oth element	Units
Vector(n) Integer (outdel) Integer (indel) Integer (n)	SP SP SP SP	R7 R6 R5	-
Output Results:		How Passed	Units
<b>Type</b> Vector(n)	Precision DP	R1 + Oth element	
Errors Detected:			
· Error #		<u>Cause</u> <u>Fixup</u>	ar i tuli. Sagara a tur

## Comments:

Registers Unsafe Across Call: Rl,R2,R4,R5,R6,R7,F0,F1.

Algorithm: Clears Fl. If outdel = 0, set to 4(HW); if indel = 0, set to 2(HW);

Loop 'length' times, adding indel to input pointer and outdel to
output pointer each time. Each loop moves current input
element to current output element.

				VV	2DN	
	HAL/	'S-FC LIBRARY	ROUTINE DES	SCRIPTION		
Source Mem	ber Name:	VV2D <b>N</b>	Size of Co	ode Area:	14	Hw
Stack Requ	irement:	O Hw	Data CSECT	Size:		O Hw
	☑ Intrinsic		D Proc	cedure		
Other Libr	ary Modules	Referenced:	NONE	<del></del>		
Primary En				lenth $\bar{n}$ .		
V1+V2	: ler emitted	code for HAL, d V2 are doub	/S construct	of the form: n vectors of 1	.ength ≠ 3	
Execution !  Input Argur		econds): 8.8	3 + 20.6n			
Туре		Precision		How Passed		Units
Vecto	or (n) or (n) ger (n)	DP DP SP		R2 →Oth elem R3 →Oth elem R5		
Output Resu						
<u>Type</u>		Precision		How Passed		Units
Vector	~ (n)	DP		$R1 \rightarrow 0^{th}$ elem	ent	
Errors Dete Error			<u>Cause</u>		Fixup	
Comments: Regi:	sters Unsafe	Across Call:	: R1,R2,R3,	R4,R5,F0,F1.		
Algorithm:	Length. Loading of	an element i	s done with	equence contro two LE instruc acies of R3 wh	ctions ins	tead

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

//S-FC LIBRARY ROUTIN		
/S-FC LIBRARY ROUTIN		
	E DESCRIPTION	
VV2D3 Size	of Code Area: 22	Hw
0 Hw Data	CSECT Size:	O Hw
ic 🔲	Procedure	
s Referenced: NONE		
IONS		
VV2D3		
	ctors.	
d code for HAL/S con V1 and V2 are double	struct of the form: precision 3-vectors.	
odules:		
coseconds): 51.4		
Precision	How Passed	<u>v</u> 1
DP DP	R2 +0th elemen	
Precision	How Passed	<u>u</u>
	Rl → Oth elemen	Ł
<b>D₽</b>		
DP <u>Cau</u>	<u>se</u>	ixup
	<u>se</u>	
	0 Hw Data ic  S Referenced: NONE  NONE  NONE  Ouble precision 3-vec  d code for HAL/S con V1 and V2 are double  coseconds): 51.4  Precision  DP  DP	O Hw Data CSECT Size:  Procedure  Referenced: NONE  NONE  Ouble precision 3-vectors.  d code for HAL/S construct of the form: V1 and V2 are double precision 3-vectors.  dodules:  Coseconds): 51.4  Precision DP DP R3 + 0th element

Registers Unsafe Across Call: R1,R2,R3,R4,F0,F1,F2,F3,F4,F5.

Algorithm: Loads F0,F2,F4 with first half of each elemnt of V2 Due to addressing Loads F1,F3,F5 with second half of each elemnt of V2 peculiarities of R3

Adds double from V1 to F0, F2, F4; Stores double into elements of result.

		VV2SN	<u></u>
HA	L/S-FC LIBRARY ROUTINE	DESCRIPTION	
	Size o	3.0	Hw .
	0 Hw Data C		O Hw
		Procedure	
X Intrins	sic L	Piocedure	
Other Library Module	es Referenced: NONE		
ENTRY POINT DESCRIPT	TIONS		
Primary Entry Name:	VV 2SN		
Primary Energ Name.	ingle precision vectors	of lenth n .	
(Also use	ed to add two matrices)		
Other Library	Modules:		
	Modules: roseconds): 8.4 + 13.	<b>6n</b>	
Execution Time (mic		6n	
Execution Time (mic		How Passed	Units
Execution Time (mic Input Arguments: Type Vector (n)	roseconds): 8.4 + 13.  Precision SP	How Passed $R2 \rightarrow 0$ th element	<u>Units</u>
Execution Time (mice Input Arguments:  Type Vector (n) Vector (n)	roseconds): 8.4 + 13.	How Passed	Units 
Input Arguments:  Type Vector (n) Vector (n) Integer (n)	roseconds): 8.4 + 13.  Precision SP SP SP SP SP	How Passed  R2 + Oth element R3 + Oth element R5	Units
Execution Time (mic Input Arguments: Type Vector (n) Vector (n)	roseconds): 8.4 + 13.  Precision SP SP	How Passed R2 + Oth element R3 + Oth element	
Input Arguments:  Type Vector (n) Vector (n) Integer (n) Output Results: Type	roseconds): 8.4 + 13.  Precision SP SP SP SP Precision	How Passed R2 + 0th element R3 + 0th element R5  How Passed R1 + 0th element	Units
Input Arguments:  Type Vector (n) Vector (n) Integer (n) Output Results: Type Vector (n)	roseconds): 8.4 + 13.  Precision SP SP SP SP Precision	How Passed R2 + Oth element R3 + Oth element R5  How Passed R1 + Oth element	<u>Unit</u>
Input Arguments:  Type Vector (n) Vector (n) Integer (n) Output Results: Type Vector (n) Errors Detected: Error #	roseconds): 8.4 + 13.  Precision SP SP SP SP SP Precision SP	How Passed R2 + Oth element R3 + Oth element R5  How Passed R1 + Oth element	Unit
Input Arguments:  Type Vector (n) Vector (n) Integer (n) Output Results: Type Vector (n)  Errors Detected: Error #	roseconds): 8.4 + 13.  Precision SP SP SP SP SP Precision SP	How Passed R2 + 0th element R3 + 0th element R5  How Passed R1 + 0th element	Unit

				VV2S	53	
	HAL/S-	FC LIBRARY	ROUTINE DES	CRIPTION		
Source Member	Name: VV2	253	Size of Co	ode Area:	1.2	Hw
Stack Require	ement:	O Hw	Data CSECT	Size:	0	Hw
	Intrinsic			edure		
Other Library	/ Modules Re	ferenced:	NONE			
ENTRY POINT I	DESCRIPTIONS					
Primary Entry	Name:	VV2S3				
Function: Add			3-vectors.			
				t of the form:		
Other L	ibrary Modul	les:				
Execution Tir	me (microsed	conds): 29.	1			
Input Argumen	nts:					
Type Vector(		Precision SP		How Passed R2 → 0th eleme	nt	Unit
Vector(		SP		R3 →0th eleme		
Output Resul	ts:					
Type Vector (	<u> </u>	Precision SP		How Passed Rl → Oth eleme	nt	<u>Unit</u>
Errors Detec	ted:					
Error			Cause		Fixup	
Comments: Regi	sters Unsaf	e Across Ca	11: R1,R2,R	3,R4,F0,F1,F2,F3	,F4,F5.	
· · · · · · · · · · · · · · · · · · ·	Loads elemen Adds elemen Stores FO.F	ts of V2 re				

DESCRIPTION  f Code Area:  SECT Size:  Procedure  ength n-vector fromes).  cruct of the form:	m another.	
Procedure  ength n-vector fromes).	0 m another.	Hw
Procedure  ength n-vector fromes).	m another.	
ength n-vector fromes).		
ength n-vector fromes).		
es). truct of the form:		
truct of the form;	f length ≠	3.
cruct of the form:	f length ≠	3.
precision vectors o	f length ≠	3.
precision vectors o	r Tengen 🗲	. J.
7n.		
How Passed R2 → Oth elemen		<u>ה</u>
R2 → Oth elemer		
R5 R5		
How Passed		<u>U</u>
$Rl \rightarrow 0^{th}$ elemen	rt.	
	Fixup	
2 74 75 70 71		
		2,R3,R4,R5,F0,F1.

on length. Load of minuend elements is done with two LE

instructions due to use of R3 as index.

	VV3D3	·	
HAL/S-FC LIBRARY ROUTINE	DESCRIPTION		
Source Member Name: VV3D3 Size o		24	Hw
tack Requirement: 0 Hw Data C		0	Hw
☐ Intrinsic ☐	Procedure		
Other Library Modules Referenced: NONE			
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: VV3D3			
Function: Subtracts two double precision ve	ectors of length 3		
Invoked by:  X Compiler emitted code for HAL/S const V1 - V2 where V1 and V2 are double pr	truct of the form: recision 3-vector		
Other Library Modules:			
Execution Time (microseconds): 55.4			
Input Arguments: Type Precision	How Passed		Uni
Vector(3)V1 DP Vector(3)V2 DP	$R2 \rightarrow 0$ th element R3 $\rightarrow 0$ th element R3 $\rightarrow 0$ th element R3		- 4 i <u>-</u>
ACCOUNTAL NO.			
Output Results:	How Passed		Uni
Type Precision	Rl O <sup>th</sup> ele	ment	
Vector (3)	in the state of th		
Errors Detected: Cause		Fixup	
Error #			
Comments:	ניש טים אמי כמי כי		
Registers Unsafe Across Call: Rl, F	الفائد الدام الفيوال في الروايل الإنابية في المعارك المعارك الدارك المعارك المعارك المعارك المعارك المعارك الم		
Algorithm: Exchange contents of R2 and R3 Load minuend elements into FO/ instruction each because of R3 Subtract subtrahend elements	/FI, F2/F3, F4/F5 us	siderati ing two	ons. LE

REPRODUCIBILITY OF THE ORIGINAL PAGE (S) POOR

			/V3SN	
		OUTINE DESCRIPTION		
Source Member Name:	VV3SN	Size of Code Area:	10	Hw
Stack Requirement:	0 <sub>Hw</sub>	Data CSECT Size:	0	Hw
X Intrin	nsic	☐ Procedure		
Other Library Modul	es Referenced: No	ne		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	VV3SN			
Function: Subtraction: (A	ts one length n si lso used to subtra	ngle precision vector matrices).	from anot	her.
		construct of the for single precision vect		α+h ≠ 3
Other Library				
	Modules:			
Other Library  Execution Time (mic	Modules:			
Other Library  Execution Time (mic  Input Arguments:  Type	Modules:	13.6n How Passe		Units
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(n)V2	Modules: roseconds): 8.4 +  Precision SP	13.6n  How Passe  R2 $\rightarrow$ 0th elem	d ment	
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(n)V2  Vector(n)V1	Modules: roseconds): 8.4 +  Precision SP SP	13.6n  How Passe  R2 $\rightarrow$ 0th elem  R3 $\rightarrow$ 0th elem	d ment	
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(n)V2	Modules: roseconds): 8.4 +  Precision SP	13.6n  How Passe  R2 $\rightarrow$ 0th elem	d ment	
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(n)V2  Vector(n)V1  Integer(n)	Modules: roseconds): 8.4 +  Precision SP SP	13.6n  How Passe  R2 $\rightarrow$ 0th elem  R3 $\rightarrow$ 0th elem  R5	d ment ment	
Other Library  Execution Time (mic  Input Arguments:  Type Vector(n)V2 Vector(n)V1 Integer(n) Output Results:  Type Vector(n)	Modules:  roseconds): 8.4 +  Precition SP SP SP SP Precision	How Passe R2 → Oth elei R3 → Oth elei R5	d ment ment	Units -
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(n)V2  Vector(n)V1  Integer(n)  Output Results:  Type	Modules:  roseconds): 8.4 +  Precition SP SP SP SP Precision	13.6n  How Passe  R2 $\rightarrow$ 0th elem  R3 $\rightarrow$ 0th elem  R5	d ment ment	<b>Uni</b>
Other Library  Execution Time (mic  Input Arguments:	Modules: roseconds): 8.4 +  Precision SP SP SP Precision SP	13.6n  How Passe  R2 $\rightarrow$ 0th elem  R3 $\rightarrow$ 0th elem  R5	d ment ment	Uni

Stack Requir	HAL/S-FC er Name: VV3S3 rement: 0	Size	NE DESCRIPTION of Code Area:		
Stack Requir	er Name: VV3S3 rement: 0	Size			
Stack Requir	rement: 0	<del></del>	or Code Area:		TT
X		West Trade		12	HW
	7 Intrinsic	nw Data	CSECT Size:	0	Hw
Other Librar	•		Procedure		
	ry Modules Refer	enced: NONE			
	DESCRIPTIONS WAS				
	ry Name: VV3S3		vectors of length	3	
Function: 50		dre breezeren	, vectors or rength		
V1-V2 W			struct of the form:		
	3)V1 Prec		How Passed R2 + 0 <sup>th</sup> elem R3 + 0 <sup>th</sup> elem	ment	<u>Uni</u>
Input Argume Type Vector(3 Vector(3	ime (microsecondents:  3) V1 3) V2 S 1ts:	s): 29.6 ision	How Passed R2 $\rightarrow$ 0 <sup>th</sup> elem R3 $\rightarrow$ 0 <sup>th</sup> elem	ment mnet	
Execution Time Input Argume Type Vector(3 Vector(3) Output Resul	ents:  Prec 3) V1 3) W2  S  Prec	s): 29.6  ision P	How Passed R2 $\rightarrow$ 0 <sup>th</sup> elem R3 $\rightarrow$ 0 <sup>th</sup> elem How Passed	ment mnet	
Input Argume Type Vector(3 Vector(3	ents:  Prec 3) V1 3) V2  S  S  1ts:  Prec (3)	s): 29.6 ision	How Passed R2 $\rightarrow$ 0 <sup>th</sup> elem R3 $\rightarrow$ 0 <sup>th</sup> elem	ment mnet	Uni - - -

				VV4DN	
	and the system of the system o	NI/C_DC TIDDADV	ROUTINE DESCRIPTION		
		AU/ S-FC DIBRARI	KOOTINE DESCRIPTION		
.1.	Source Member Name:	VV4DN	Size of Code Area:	8	Hw
	Stack Requirement:	0 Hw	Data CSECT Size:	0	Hw
	X Intrins	sic	☐ Procedure		
	Other Library Module	es Referenced:	None		
	ENTRY POINT DESCRIPT	rions			
	Primary Entry Name:				
			of a double precisio	n length n ve	ctor by
		precision scal to multiply mat			
	Invoked by:	co marcipily mac	in Di Dourari,		
		d ando for Wat	عاوا من با		
	- I COMPTTOT CHITCH	ed code for unity	'S construct of the f	Orm:	
	V S, where V		cision vector of leng		
	V S, where V	is a double pre is a double pre	cision vector of leng		
	V S, where V	is a double pre is a double pre	cision vector of leng		
	V S, where V S Other Library M	is a double pre is a double pre Modules:	cision vector of leng		
	V S, where V	is a double pre is a double pre Modules:	cision vector of leng		
	V S, where V S Other Library M Execution Time (micro	is a double pre is a double pre Modules:	cision vector of leng		
	V S, where V S Other Library M	is a double pre is a double pre Modules:	cision vector of leng cision scalar. 0 + 23.4n	th ≠ 3, and	Uni
	V S, where V S Other Library M  Execution Time (michae) Input Arguments:	is a double pre is a double pre Modules: roseconds): 7.  Precision DP	cision vector of length eision scalar. $0 + 23.4n$ $\frac{\text{How Pas}}{\text{R2} \rightarrow 0^{\text{th}} \text{ el}}$	th ≠ 3, and	<u>Uni</u>
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP	cision vector of length cision scalar. $0 + 23.4n$ $\frac{\text{How Pas}}{\text{R2} \rightarrow 0^{\text{th}} \text{ el}}$	th ≠ 3, and	<u>Uni</u>
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n)	is a double pre is a double pre Modules: roseconds): 7.  Precision DP	cision vector of length eision scalar. $0 + 23.4n$ $\frac{\text{How Pas}}{\text{R2} \rightarrow 0^{\text{th}} \text{ el}}$	th ≠ 3, and	<u>Uni</u>
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n) Output Results:	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP  DP  SP	cision vector of lenguision scalar. $0 + 23.4n$ $R2 \rightarrow 0^{th} el$ $R5$	th ≠ 3, and  ssed ement	
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n)	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP	cision vector of length cision scalar. $0 + 23.4n$ $\frac{\text{How Pas}}{\text{R2} \rightarrow 0^{\text{th}} \text{ el}}$	th ≠ 3, and  ssed ement	Uni-
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n) Output Results: Type	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP  SP  Precision	cision vector of lenguision scalar. $0 + 23.4n$ How Pas  R2 $\rightarrow 0^{th}$ e1  F0  R5	th ≠ 3, and  ssed ement	
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n) Output Results: Type Vector (n)	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP  SP  Precision	cision vector of lenguision scalar. $0 + 23.4n$ How Pas  R2 $\rightarrow 0^{th}$ e1  F0  R5	th ≠ 3, and  ssed ement	
	V S, where V S Other Library M  Execution Time (mich Input Arguments: Type Vector (n) Scalar Integer (n) Output Results: Type Vector (n)  Errors Detected:	is a double pre is a double pre Modules:  roseconds): 7.  Precision  DP  DP  SP  Precision	cision vector of lenguision scalar. $0 + 23.4n$ $R2 \rightarrow 0^{th} el$ $F0$ $R5$ $R1 \rightarrow 0^{th} el$	th ≠ 3, and  ssed ement ssed ement	

multiply, store for each element.

		VV4D3	<del></del>
ТАН	/S-FC LIBRARY ROUTI	NE DESCRIPTION	
Source Member Name: _	VV4D3 Size	e of Code Area: 18	Hw
Stack Requirement:	0 Hw Data	CSECT Size: 0	Hw
▼ Intrinsi	La 🗀	Procedure	
Other Library Modules	Referenced: No.	ne.	
ENTRY POINT DESCRIPT	IONS		
Primary Entry Name:	VV4D3		
	olies each element o	f a double precision vector ar.	of lengt
Invoked by:  Compiler emitted	d code for HAL/S con	nstruct of the form:	
V S, where V i	is a length 3 double is a double precisio	precision vector, and n scalar.	
Other Library Mo	and the first of the control of the		
Execution Time (micro	oseconds): 68.4		
Input Arguments: Type	Precision	How Passed	TT-m -
And the second second		<u>0</u> 4	Uni
Scalar	DP		on
Vector (3)	DP	$R2 \rightarrow 0^{\text{th}} \text{ element}$	<u>on</u>
Vector (3)			<u>on</u>
<del> </del>			
Vector (3) Output Results:	DP	$R2 \rightarrow 0$ <sup>th</sup> element	<u>Un</u>
Vector (3) Output Results: Type	DP Precision	$R2 \rightarrow 0^{th}$ element  How Passed	

			VV4SN	
HA	L/S-FC LIBRARY	ROUTINE DESCRIPTION	1	
Source Member Name:	vv4sn	Size of Code Area	. 8	_ Hw
Stack Requirement: _				Hw
▼ Intrins		Procedure		
Other Library Module	s Referenced: _	None.		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	VV4SN			
		ingle precision vec	tor by a single	1 32
precision sca	lar. (Also	ised to multiply ma	trix by scalar).	
Invoked by:				
Compiler emitte	d code for HAL/	'S construct of the	form:	
$\overline{V}$ s, where $V$	v is a single pr s is a si <b>n</b> gle pr	ecision vector of l	ength ≠ 3, and	
Other Library M				
C Other mining				
Execution Time (micr	roseconds):	<b>5</b> 0 1 1 4 0-		
DVCCACTON TTWO (1117)		1.0 + 14.00	to para organization in	
		7.0 + 14.0n		
Input Arguments:			Paggad.	Uni
Type	Precision SP	How I		<u>Uni</u>
<u>Type</u> Scalar Vector(n)	Precision SP SP	How I	O th element	<u>Uni</u>
Type Scalar Vector(n) Integer(n)	Precision SP	$\frac{\text{How I}}{\text{F}}$ R2 \rightarrow 0	O th element	<u>Uni</u>
<u>Type</u> Scalar Vector(n)	Precision SP SP	How I F R2 → 0 R How 1	O th element 5 Passed	
Type Scalar Vector(n) Integer(n) Output Results:	Precision SP SP SP SP	How I F R2 → 0 R How 1	O th element 5	Uni Uni
Type Scalar Vector(n) Integer(n) Output Results: Type	Precision SP SP SP SP Precision	$\frac{\text{How I}}{\text{R2} \to 0}$ $\frac{\text{How I}}{\text{R}}$ $\frac{\text{How I}}{\text{R1} \to 0}$	Oth element 5 Passed th element	
Type scalar Vector(n) Integer(n) Output Results: Type Vector(n)	Precision SP SP SP SP Precision	How I F R2 → 0 R How 1	O th element 5 Passed	
Type Scalar Vector(n) Integer(n) Output Results: Type Vector(n) Errors Detected:	Precision SP SP SP SP Precision	$\frac{\text{How I}}{\text{R2} \to 0}$ $\frac{\text{How I}}{\text{R}}$ $\frac{\text{How I}}{\text{R1} \to 0}$	Oth element 5 Passed th element	
Type Scalar Vector(n) Integer(n) Output Results: Type Vector(n) Errors Detected:	Precision SP SP SP SP Precision	$\frac{\text{How I}}{\text{R2} \to 0}$ $\frac{\text{How I}}{\text{R}}$ $\frac{\text{How I}}{\text{R1} \to 0}$	Oth element 5 Passed th element	
Type Scalar Vector(n) Integer(n) Output Results: Type Vector(n) Errors Detected: Error #  Comments:	Precision SP SP SP SP SP SP	$\frac{\text{How I}}{\text{R2} \to 0}$ $\frac{\text{How I}}{\text{R}}$ $\frac{\text{How I}}{\text{R1} \to 0}$	Oth element  Seassed  th element  Fixup	

			VV4S3
НА	./s-FC LIBRARY R	OUTINE DESCRIPTION	
Source Member Name:	VV4S3	Size of Code Area: _	12 Hw
Stack Requirement:	O Hw	Data CSECT Size:	O· Hw
X Intrins	ic	Procedure	
Other Library Module	s Referenced:	None	
ENTRY POINT DESCRIPT	IONS		
Primary Entry Name:	VV4s3		
Function: Multipli	les each element cision scalar.	of a single precisio	n 3-vector by a
V S where V	d code for HAL/ is a single pre is a single pre	S construct of the focision 3-vector, and cision scalar.	)rm;
Other Library M			
			the state of the s
Execution Time (micr	coseconds): 38.	<b>4</b>	
Execution Time (micronic) Input Arguments:			sed [
	Precision SP	How Pas	<del> </del>
Input Arguments: Type	Precision	How Pas	sed [
Input Arguments:  Type Scalar Vector(3)  Output Results:	Precision SP SP	How Pas F0 R2 → 0 <sup>tl</sup>	n element
Input Arguments:  Type Scalar Vector(3)	Precision SP	$\frac{\text{How Pas}}{\text{F0}}$ $\text{R2} \rightarrow 0^{\text{tl}}$ $\text{How Pas}$	n element
Input Arguments:  Type Scalar Vector(3)  Output Results: Type	Precision SP SP Precision	How Pas $ \begin{array}{c} \text{F0} \\ \text{R2} \rightarrow 0^{\text{tl}} \end{array} $ How Pas $ \begin{array}{c} \text{R1} \rightarrow 0^{\text{tl}} \end{array} $	element sed !
Input Arguments:  Type Scalar Vector(3)  Output Results:  Type Vector(3)	Precision SP SP Precision	$\frac{\text{How Pas}}{\text{F0}}$ $\text{R2} \rightarrow 0^{\text{tl}}$ $\text{How Pas}$	n element sed
Input Arguments:  Type Scalar Vector(3)  Output Results:  Type Vector(3)  Errors Detected:	Precision SP SP Precision	How Pas $ \begin{array}{c} \text{F0} \\ \text{R2} \rightarrow 0^{\text{tl}} \end{array} $ How Pas $ \begin{array}{c} \text{R1} \rightarrow 0^{\text{tl}} \end{array} $	element sed !
Input Arguments:  Type Scalar Vector(3)  Output Results: Type Vector(3)  Errors Detected: Error #	Precision SP SP SP  Precision SP	How Pas $ \begin{array}{c} \text{F0} \\ \text{R2} \rightarrow 0^{\text{tl}} \end{array} $ How Pas $ \begin{array}{c} \text{R1} \rightarrow 0^{\text{tl}} \end{array} $	element sed !

REPRODUCIBILITY OF THE ORIGINAL PAGE IS FOOD

			VV5DN		
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTI	ON		
Source Member Name		Size of Code Are		L6	Hw
Stack Requirement:		Data CSECT Size:		2	Hw
🔀 Intri		☐ Procedure			
Other Library Modu	les Referenced:	None.			
ENTRY POINT DESCRI	PTIONS				
Primary Entry Name	• VV5DN				
Function: Divides precision sca	a double precisi lar. (Also u	on vector of lengt sed to divide matr	th n by a c ix by sca	louble Lar).	
Invoked by:  X Compiler emit	ted code for HAL	/S construct of th	ne form:		
$\overline{V}/S$ , where $\square$ Other Library	S is a double pre	cision vector of I	Length ≠ 3	, and	
Execution Time (mi	.croseconds):	37.0 + 24.2n			
Input Arguments:					<b>77</b>
Type Scalar Vector (n) Integer (n)	Precision  DP DP SP	$R2 \rightarrow 0$	Passed FO Oth elemen R5	t	Units - -
Output Results:	Provision	How	Passed		Units
<u>Type</u> Vector (n)	<u>Precision</u> DP	and the second s	Oth elemen	t	-
Errors Detected:					
Error #		Cause		Fixup	
25	Scalar argume	nt is zero. Sto	re origina	1 vector	as result
Comments:					
Registers Unsa	fe Across Call:	R1,R2,R4,R5,F0,F1,	F2,F3.		
1.79	s and then use BC	preset quotient to TB loop to count d 1/S), store sequen	own 'lengt	h' perfor	ming

Garage Markey Namos	777.ED2	Size of Code Area:	26	Hw
Source Member Name:	VV3D3			
Stack Requirement:	O Hw	Data CSECT Size:	2	_ Hw
X Intrin	sic	☐ Procedure		
Other Library Module	es Referenced:	None		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	VV5D3			
Invoked by:		/a senset west of the for	<b>~</b>	
X Compiler emitt	V is a double pr S is a double pr Modules:		<b>m:</b>	
<pre> X Compiler emitt  V/S, where v  Other Library  Execution Time (mice)  X Compiler emitt  V/S, where v  Compiler emitt  V/</pre>	V is a double pr S is a double pr Modules:	ecision 3-vector, and ecision scalar.		
X Compiler emitt  V/S, where V  Other Library  Execution Time (mic	V is a double proseconds):  Precision	ecision 3-vector, and ecision scalar.		<u>Units</u>
X Compiler emitt  V/S, where V  Other Library  Execution Time (mic	V is a double pr S is a double pr Modules: croseconds): 98	ecision 3-vector, and ecision scalar.	ed.	<u>Units</u>
X Compiler emitt  V/S, where V  Other Library  Execution Time (mic	V is a double property of the	recision 3-vector, and recision scalar.  3.4  How Passe FO R2 + 0th electric scalar.	<u>ed</u>	
Compiler emitt  V/S, where V  Other Library  Execution Time (mic  Input Arguments:  Type Scalar Vector(3)	V is a double property of the	ecision 3-vector, and ecision scalar.  3.4  How Passer	ed ement	Units - - - -

Registers Unsafe Across Call: R1,R2,R4,F0,F1,F2,F3,F4,F5.

Test FO; if zero, send error and set quotient to 1; Algorithm: Otherwise, quotient l/arg is calculated and then used in simple Load, multiply, store sequence for each element.

		·	VV5SN	•
HZ	AL/S-FC LIBRARY ROU	TINE DESCRIPTION		
Source Member Name:		ze of Code Area:	14	Hw
Stack Requirement: _	<u>0</u> Hw Da	ta CSECT Size:	2	Hw
☑ Intrins	sic	<pre>Procedure</pre>		
Other Library Module	es Referenced: No	one.		
ENTRY POINT DESCRIPT	CIONS			
Primary Entry Name:	VV5sn			
Function:				
Divides single pred	ision vector of ler	ngth n by single p	recision scal	ar.
		divide matrix by		
Invoked by:			· ·	
Z Compiler emitte	ed code for HAL/S c	onstruct of the f	orm:	
$\overline{V}/S$ , where V i	s a single precisio	on vector of lengt	$h \neq 3$ , and	
S i	s a single precisio	on scalar.		
Other Library M	lodules:			
<b>Execution Time (micr</b>	oseconds): 72 + 1			
	05000	8.0n		
		8.0n		
Input Arguments:				
Type	Precision	How Pas	sed .	Uni
<u>Type</u> Scalar	Precision SP	How Pas		<u>Uni</u>
<u>Type</u> Scalar Vector (n)	Precision	How Pas		Uni:
Type Scalar Vector (n) Integer (n)	Precision SP SP			Uni
<u>Type</u> Scalar Vector (n)	Precision SP SP		element	
Type Scalar Vector (n) Integer (n) Output Results:	Precision SP SP SP SP	$ \frac{\text{How Pas}}{\text{F0}} $ R2 \rightarrow 0th R5	element sed	
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n)	Precision SP SP SP SP Precision	How Pas F0 R2 → 0th R5 How Pas	element sed	
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected:	Precision SP SP SP SP Precision SP	How Pas F0 R2 → 0th R5 How Pas R1 → 0th	element sed element	Unit
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected: Error #	Precision SP SP SP Precision SP	How Pas F0 R2 → 0th R5  How Pas R1 → 0th	element sed element Fixup	Uni
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected:	Precision SP SP SP SP Precision SP	How Pas F0 R2 → 0th R5  How Pas R1 → 0th	element  sed element <u>Fixup</u> Store origina	Unit
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected: Error #	Precision SP SP SP Precision SP	How Pas F0 R2 → 0th R5  How Pas R1 → 0th	element sed element Fixup	Unit
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected: Error #	Precision SP SP SP Precision SP	How Pas F0 R2 → 0th R5  How Pas R1 → 0th	element  sed element <u>Fixup</u> Store origina	Unit
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected: Error # 25 . Comments:	Precision SP SP SP Precision SP	How Pas $ \begin{array}{c}                                     $	element  sed element  Fixup  Store origina result.	Uni
Type Scalar Vector (n) Integer (n) Output Results: Type Vector(n) Errors Detected: Error # 25 . Comments: Registers Unsafe	Precision SP SP SP Precision SP Can	How Pas $R2 \rightarrow 0 \text{th}$ $R5$ How Pas $R1 \rightarrow 0 \text{th}$ use at is zero.	element  sed element  Fixup  Store origina result.	Uni Uni

				<u> </u>	
			VV	7583	
 HA	L/S-FC LIBRARY	ROUTINE DESCI	RIPTION		
 Source Member Name:	VV5s3	Size of Code	Area:	18	_ Hw
Stack Requirement:	O Hw	Data CSECT S	Size:	2	Hw
☑ Intrins	ic	☐ Proced	ure		
Other Library Module	s Referenced:	None.			
ENTRY POINT DESCRIPT	IONS				
Primary Entry Name:	VV5s3				
Function: Divide e single p	ach element of recision scalar		ision vector	of length	3 by a
Invoked by: X Compiler emitte	d code for HAL,	/S construct o	of the form:		
$\overline{V}/S$ , where V	is a single pre	cision 3-vecto	or, and		
S Other Library M	is a single pre odules:	cision scalar			
Execution Time (micr	oseconds): 50	.6			
Input Arguments: Type	Precision		How Passed		Units
Scalar	SP		FO		-
Vector(3)	SP	R2	→ 0th eleme	ent	-
Output Results:					
Type	Precision		How Passed		Units
Vector(3)	SP	Rl	→ 0th eleme	ent	
Errors Detected:					
Error #		Cause		Fixup	
25	Scalar argu	ment is zero.	Store or	riginal vect	or as
				result.	
Comments:					
Registers Unsafe A	Across Call: R	L,R2,R4,F0,F1,	F2,F3.		
1125022011111	0; if zero, set				е

			VV6DN	I
HAL	/S-FC LIBRARY	ROUTINE DI	ESCRIPTION	
 Source Member Name: _	VV6DN	Size of (	Code Area:	12 Hw
Stack Requirement:	O Hw	Data CSEC	CT Size:	O Hw
🖳 Intrinsi	c	☐ Pro	ocedure	
Other Library Modules	Referenced:	NONE		
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name:	VV6DN			
Function: Forms dot	product of	two double	precision length	n vectors.
Invoked by:  Compiler emitted	code for HAL	/S construc	t of the form:	
			on vectors of lend	gth n, $n \neq 3$ .
Other Library Mo	dules:			
Execution Time (microstant) Input Arguments:		4 + 25.4n		
<u>Type</u> Vector (n)	Precision		How Passed	Units
Vector (n) Vector (n) Integer (n)	DP DP SP		$R2 \rightarrow 0$ <sup>th</sup> element $R3 \rightarrow 0$ <sup>th</sup> element R5	
Output Results:				
Type	Precision		How Passed	Units
Scalar	DP		FO	
Errors Detected:				
Error #		Cause		ixup
Comments: Registers Unsafe Ac	ross Call: R	1,R2,R3,R4,	R5,F0,F1,F2,F3.	
Algorithm: Loads R3 ir performs:	nto RL for add	ressability	v advantages	
n Σ i=1	$ extsf{V1}_{ extsf{i}}$ $ extsf{V2}_{ extsf{i}}$ by lo	ops countir	ng down n;	
	oads Vl mult	iplies by V	$^{\prime 2}$ i and add to	

		_	VV6D3	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member	Name: VV6D3	Size of Code Area:	16	Hw
Stack Requirem	ment: 0 Hw	Data CSECT Size:		O Hw
<b>2</b> 3 1	Intrinsic	Procedure		
Other Library	Modules Referenced:	NONE		
ENTRY POINT DE	Name: VV6D3			
Function: Fo	rms dot product of 2	double precision 3-v	ectors.	
Invoked by:  Compiler	emitted code for HAL	/S construct of the	form:	
V1.V2 W	here V1 and V2 are do	uble precision 3-vec	tors.	
Execution Tim	e (microseconds): 7	1.8		
Input Argumen Type	ts: Precision	How Pa	assed	Units
Vector(3 Vector(3		R2 →0 <sup>t</sup> R3 →0 <sup>t</sup>	h element h el <b>e</b> ment	
Output Result <u>Type</u> Scalar	s: Precision DP	How Pa	<del></del>	<u>Units</u>
Errors Detect		<u>Cause</u>	Fixup	
	Unsafe Across Call:			
		via straight line c		
	accummulating result			

REPRODUCIBILITY OF THE ORIGINAL PAGE IS In the second

		VV	5sn
на	L/S-FC LIBRARY ROU	TINE DESCRIPTION	
Source Member Name:		ze of Code Area:	12 Hw
Stack Requirement: _	0 Hw Da	ta CSECT Size:	O Hw
▼ Intrins	ic	Procedure	
Other Library Module	s Referenced: NO	NE	
ENTRY POINT DESCRIPT			
Primary Entry Name:	VV6SN	*	
Function: Forms dot	product of two le	ength n single precision	vectors.
Invoked by:    X   Compiler emitter     V1.V2   where V1     Other Library     Execution Time (michae)     Input Arguments:   Type     Vector (n)     Vector (n)     Integer (n)     Output Results:	and V2 are single	econstruct of the form:  precision n-vectors, n  How Passed  R2 + 0th eleme R3 + 0th eleme R5	Units ent - ent -
Type	Precision	How Passed	<u>Units</u>
Scalar	SP	FO	
Errors Detected: Error #	<u>S</u>	Cause	Fixup
Comments:			
		,R2,R3,R4,F5,F0,F1,F2,F	
perform	s: n Σ V1 <sub>i</sub> V2 <sub>i</sub> by i=1	ability advantages  a loop counting down n	
	ss loads $Vl_1$ , mult	ciplies by V2 <sub>1</sub> and adds	

		VV6S3	
		***	
Н	AL/S-FC LIBRARY ROUTIN	E DESCRIPTION	
Source Member Name:	VV6S3 Size	of Code Area: 10	Hw .
Stack Requirement:	0 Hw Data	CSECT Size:	O Hw
Intrins	sic 🔲	Procedure	
Other Library Module	es Referenced: NONE		
ENTRY POINT DESCRIPT	PIONS		
Primary Entry Name:	VV6S3		
Function: Forms do	ot product of two sing	le precision 3-vectors.	
Invoked by:			
	ed code for HAL/S cons and V2 are single pre		
Other Library M	Modules:		
Execution Time (mic	roseconds): 41.8		
Input Arguments:	Precision	How Passed	Units
<u>Type</u> Vector (3)	SP	R2 →oth element	<u> </u>
Vector (3)	SP	R3 →0 <sup>th</sup> element	
Output Results:			
Type	Precision	How Passed	Units
Scalar	SP	FO	
Errors Detected:			
Error #	<u>Cause</u>	<u>Fixu</u>	
Comments:			
	Across Call: R2,R3,R	4,F0,F1,F2,F3.	
Algorithm: Calculat	es V1 V2 +V1 V2 +	V1 V2 via direct code r	o loons
		V1 <sub>3</sub> V2 via direct code, r	
accummul	lating result in FO.		

5-224

1

		<u> </u>	VV 7DN	
нат	/S-FC T.TRRARY	ROUTINE DESCRIPTION		
	VV7DN	Size of Code Area:	8	Hw
Source Member Name: _			0	— Hw
Stack Requirement:	0 <sub>Hw</sub>	Data CSECT Size:		IIW
☑ Intrinsi	.c	Procedure		
Other Library Modules	Referenced:	None.		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	VV7DN			
Function: Vector neg	ate, double pr	ecision, length n.		
	l to negate mat			
Invoked by:				
Compiler emitted	d code for HAL,	/S construct of the f	OTIII;	
-V where V is	a double preci	sion vector of length	n n, n≠3.	
-V, where V is	a double preci	sion vector of length	ı n, n≠3.	
-V, where V is		sion vector of length	n n, n≠3.	
Other Library Mo	odules:	sion vector of length	n n, n≠3.	
	odules:	sion vector of length	n n, n≠3.	
Other Library Mo	odules:	sion vector of length	n n, n≠3.	
Other Library Mo	odules:	sion vector of length ) + 11.4n  How Pas	n n, n≠3.	Unit
Other Library Mo  Execution Time (micro  Input Arguments:  Type  Vector (n)	odules: oseconds): 7.0  Precision DP	sion vector of length	ssed element	Unit
Other Library Mo Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)	odules: oseconds): 7.0	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th	ssed element	<u>Unit</u>
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (n) Integer (n)  Output Results:	odules: oseconds): 7.0  Precision DP SP	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th	ssed element	Unit - - - - - -
Other Library Mo Execution Time (micro Input Arguments:  Type Vector (n) Integer (n)	odules: oseconds): 7.0  Precision DP	sion vector of length $0 + 11.4n$ $0 + 11$	ssed element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (n) Integer (n)  Output Results: Type	odules: oseconds): 7.0  Precision DP SP  Precision	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th R. How Pas	ssed element ssed element	
Other Library Mo  Execution Time (micro  Input Arguments:  Type Vector (n) Integer (n)  Output Results:  Type Vector(n)	odules: oseconds): 7.0  Precision DP SP  Precision	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th R. How Pas	ssed element	
Other Library Mc Execution Time (micro Input Arguments:	odules: oseconds): 7.0  Precision DP SP  Precision	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th R1  How Pas R1 $\rightarrow$ 0th	ssed element ssed element	
Other Library Mc Execution Time (micro Input Arguments:	odules: oseconds): 7.0  Precision DP SP  Precision	sion vector of length $0 + 11.4n$ How Pas R2 $\rightarrow$ 0th R1  How Pas R1 $\rightarrow$ 0th	ssed element ssed element	

5-225

			VV <b>7</b> D	3	·
HAL/S	FC LIBRARY	ROUTINE DESCR	IPTION		
Source Member Name:V	V7D3	Size of Code	Area: 18	3	Hw
Stack Requirement:	O Hw	Data CSECT S	ize:	0	Hw
X Intrinsic		☐ Proced	ure		
Other Library Modules I	Referenced:	None	r de la companya de l		
ENTRY POINT DESCRIPTION	<u>vs</u> vv7d3				
FIIMALY Effects Remes					
Function: Vector negat	e, double pr	ecision for ve	ctors of le	ngth 3.	
Invoked by:  \[ \overline{\text{X}} \] Compiler emitted of \[ -\overline{V}, \] where V is					
Other Library Mod	ıles:				
Execution Time (micros	econds): 32.	4			
Input Arguments:  Type Vector(3)	Precision DP		How Passed R2 → 0 <sup>th</sup> el	.ement	Units -
Output Results:			Now Decod		Units
Type Vector(3)	Precision DP		How Passed Rl → Oth el		Unites
Errors Detected: Error #		<u>Cause</u>		<u>Fixup</u>	
Comments: Registers Unsafe Acr	oss Call: R	1,R2,R4,F0,F1,	,F2,F3,F4,F5		
		sequence, no l negates, 3 st			

		VV7SN	<del></del>
en de la companya de La companya de la co	AL/S-FC LIBRARY ROUTI	NE DESCRIPTION	
	and the second second		
Source Member Name:	VV7SN Size	of Code Area: 8	Hw
Stack Requirement:	0 Hw Data	CSECT Size: 0	Hw
x Intrins	sic 🔲	Procedure	
Other Library Module	es Referenced: <u>None</u>		
ENTRY POINT DESCRIP	I'IONS		
Primary Entry Name:	VV7SN		
Function:			
Vector :	negate, single precist sed to negate matrices		
	sea to negate materials.		
Invoked by:  X Compiler emitte	ed code for HAL/S con	struct of the form:	
X Compiler emitte		struct of the form: vector of length n, n #3.	
Compiler emitted -V, where V is Other Library	s a single precision	vector of length n, n#3.	
Compiler emitter -V, where V is Other Library Execution Time (mic.	s a single precision v	vector of length n, n#3.	
Compiler emitted -V, where V is Other Library	s a single precision v	vector of length n, n#3.	
Compiler emitted -V, where V is Other Library Execution Time (michigan)	s a single precision wondules: roseconds): 7.0 + 9.0	vector of length n, n≠3.	<u>Un</u>
Compiler emitted -V, where V is Other Library Execution Time (miching Input Arguments:  Type	s a single precision of Modules: roseconds): 7.0 + 9.0  Precision	vector of length n, n≠3. On How Passed	<u>Un</u>
Compiler emitted -V, where V is Other Library I Execution Time (mice Input Arguments:  Type Vector (n)	s a single precision of Modules:  roseconds): 7.0 + 9.0  Precision  SP	Prector of length n, $n \neq 3$ .  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R5	
Compiler emitted variables and the compiler emitted variables and the compiler emitted variables and the compiler emitted variables.  Input Arguments:  Type  Vector (n)  Integer (n)  Output Results:  Type	Modules: roseconds): 7.0 + 9.0  Precision SP SP Precision	On  How Passed  R2 → 0 <sup>th</sup> element  R5  How Passed	
Compiler emitted -V, where V is the other Library I will be seen to the control of the control o	Modules: roseconds): 7.0 + 9.0  Precision SP SP	Prector of length n, $n \neq 3$ .  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R5	<u>Un</u>
Compiler emitted -V, where V is the order Library Defector Time (mice of the control of the cont	Modules: roseconds): 7.0 + 9.0  Precision SP SP Precision	On  How Passed  R2 → 0 <sup>th</sup> element  R5  How Passed  R1 → 0 <sup>th</sup> element	
Compiler emitted TV, where V is the Total Control of the Library I will be a secution Time (mich the Type Vector (n) Integer (n)  Output Results:  Type Vector (n)	Modules: roseconds): 7.0 + 9.0  Precision SP SP Precision	On  How Passed  R2 → 0 <sup>th</sup> element  R5  How Passed  Al → 0 <sup>th</sup> element	
Compiler emitted —V, where V is —V, where V is —V is —	Modules: roseconds): 7.0 + 9.0  Precision SP SP Precision SP	On  How Passed  R2 → 0 <sup>th</sup> element  R5  How Passed  Al → 0 <sup>th</sup> element	

negate, store sequence on current vector element.

		vv7s3	
	HAL/S-FC LIBRARY ROUT	TINE DESCRIPTION	
Source Member Name	:	ze of Code Area: 12	Hw
Stack Requirement:	0 Hw Da	ta CSECT Size: 0	Hw
👿 Intri	nsic	Procedure	
Other Library Modu	les Referenced: Non	e.	
ENTRY POINT DESCRI	PTIONS		
Primary Entry Name	.: VV7S3		
Function:			
Vector negate, si	ngle precision for ve	ctors of length 3,	
Invoked by:			
,		onstruct of the form:	
-V, where	V is a single precisi	on 3-vector.	
Other Library	Modules:		
Execution Time (mi	croseconds): 23.4		
Input Arguments:			
Туре	Precision	How Passed	Units
Vector(3)	S.P	$R2 \rightarrow 0$ <sup>th</sup> element	
Output Results: Type	Precision	How Passed	Units
	SP	$R1 \rightarrow 0^{th}$ element	
Vector (3)			
Vector (3) Errors Detected:			
		use <u>Fi</u> x	nup
Errors Detected:		use <u>Fi</u>	Rup
Errors Detected: Error #		use Fix	aup
Errors Detected:	<u>ca</u>	use Fix	np
Errors Detected: Error #  Comments: Registers Unsaf	<u>ca</u>	2,R4,F0,F1,F2,F3,F4,F5.	Rup

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POUR

		<u>_ vv</u> s	BD3
АН	L/S-FC LIBRARY ROUT	INE DESCRIPTION	
Source Member Name:	VV8D3Siz	e of Code Area:	12 Hw
Stack Requirement: _	O Hw Dat	a CSECT Size:	O Hw
👿 Intrins	ic	] Procedure	
Other Library Module	s Referenced: Non	e	
ENTRY POINT DESCRIPT	IONS		
Primary Entry Name:	VV8D3		
	two double precisi	on 3-vectors	
and the second s	where X and Y are	double precision 3-	vectors.
Other Library M	odules:		
	nodules:		if X≠Y where n =
Other Library M	nodules:	X=Y; 16.2n + 24.6	if X≠Y where n = air of elements). 1 Un ment
Other Library M  Execution Time (micr  Input Arguments:  Type Vector(3)	odules: coseconds): 59.0 if 3 -(index of  Precision DP	X=Y; 16.2n + 24.6 last non-matching parameters $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}}}$ elements	if X≠Y where n = air of elements).  Unment ment
Other Library M  Execution Time (micr  Input Arguments:  Type  Vector(3)  Vector (3)  Output Results:  Type	odules: coseconds): 59.0 if 3 -(index of  Precision DP DP	X=Y; 16.2n + 24.6 last non-matching particles and the passed R2 + 0 <sup>th</sup> election How Passed condition	if X≠Y where n = air of elements).  Unment ment

Algorithm: Loads a literal 3 into R5, then drops into VV8DN code.

VV8D3	
-------	--

Secondary En	try Name:	AASDN	——————————————————————————————————————			
Function:	Compares t	wo double pre	ecision vecto	rs of length n		
	(Also used	to compare n	matrices),		•	
Invoked by:						
			S construct double precis	of the form: ion vectors of	length n,	n≠3.
Other 1	ibrary modu	ıles:				
Execution Ti	me (micros	econds): 16.2 m = n-(index	2n + 18.0 if c of last non	X=Y; 16.2m + 22 -matching pair	2.2 if X≠Y, of element	where s)
Input Argume	ents:					
<u>Type</u> Vector Vector Integer	(n)	DP DP DP SP		How Passed  R2 $\rightarrow$ 0th ele  R3 $\rightarrow$ 0th ele  R5		Units - -
Output Resul	lts:	Precision		How Passed		<u>Units</u>
equal/un	equal			Condition cod	e.	,
Errors Detec			Cause		Fixup	
•						

Registers Unsafe Across Call: R1,R2,R3,R4,R5,F0,F1.

Algorithm:

Loads R3 into R1 for better addressability. Loops, counting down 'size', each pass compares values of one element of each vector. When first non-compare occurs, branch to return point is taken, exiting loop.

Condition code is set based upon whether count down loop reaches 0. Condition code of 00 indicates equality, 01 indicates unequality.

			VV8S3	
HA	L/S-FC LIBRARY	ROUTINE DESCRIPTION	N	*
Source Member Name:	VV8S3	Size of Code Area	: 12	Hw
 Stack Requirement:	0 Hw	Data CSECT Size:	0	Hw
☐ Intrins	ic	☐ Procedure		
Other Library Module	s Referenced:	None.		
ENTRY POINT DESCRIPT	IONS			•
Primary Entry Name:	vv8s3			
Function: Compares	two single pre	ecision vectors of	length 3.	
Invoked by:  Compiler_emitte	d code for HAL	/S construct of the	form:	
$\mathbf{TF} X = Y \dots, w$	here X and Y and	re single precision	3-vectors.	
Other Library M	odules:			
Execution Time (micr	oseconds): 42. 4 -(index o	8 if X=Y; 10.8n + 8 f last non-matching	3.4 if X≠Y, where y pair of elemen	e n = ts).
Input Arguments:				
Type Vector (3)	Precision SP		Passed h element	<u>Units</u>
Vector (3)	SP		h element	
Output Results:				
Type equal/not equal	Precision -		eassed ion Code.	Units
Errors Detected:				
Error #		Cause	Fixup	
Comments:				
Registers Unsaf	e Across Call:	R1,R2,R3,R4,R5,F0	<b>,F1.</b>	
Algonithm. Loads a	literal 3 into	o D5 Then falle i	nto WASN routir	16

5-231

Secondary Entry Name: _	VV8SN		
Function: Compares t	wo single precisi	on vectors of length n.	
	lso used to compar		
Invoked by:			
Compiler emitted of $X = X = X$ , where $X = X = X$	code for HAL/S conners X and Y are s length n, n≠3	ingle precision vectors of	
Other library mod	ules:		
Execution Time (micros	econds): 10.8n +	8.0 if X=Y; 10.8m + 6.0 if X\neq Y	, where
m =	n -(index of last	non-matching pair of elements	) +1.
Input Arguments:			
Type	Precision	How Passed $R2 \rightarrow 0$ th element	Units
Vector (n)	SP SP	$R2 \rightarrow 0$ th element	_
Vector (n) Integer (n)	SP	R5	r in e
Output Results: Type	Precision	How Passed	Units
		Condition Code.	_
equal/not equal		Cougratou Code.	
Errors Detected: Error #	<u>Ca</u> ı	ıse Fixup	

Registers Unsafe Across Call: R1,R2,R3,R4,R5,F0,F1.

Algorithm: Loads R3 into R1 for better addressability. Loops counting down 'size', each pass compares values of one element of each vector. When first non-compare occurs, branch to return point is taken, exiting loop. Condition code is set based upon whether count down loop reaches 0. Condition code of 00 indicates equality, 01 indicates inequality.

					VV10D3	
; ;	HAL	/s-FC LIBRARY	ROUTINE I	DESCRIPTION		
	Source Member Name:	VV10D3	Size of	Code Area:	56	Hw Hw
* .	Stack Requirement:	20 Hw	Data CSI	ECT Size:	2	Hw
	☐ Intrinsi	c	X Pro	ocedure		
	Other Library Modules	Referenced:	DSQR	T, VVODN		
	ENTRY POINT DESCRIPTI	ONS				
	Primary Entry Name: _	VV10D3			•	
			length 3	for input 3-ve	ector in	
	double pr	ecision.				
	Invoked by:  X Compiler emitted	code for HAL	/S constr	uct of the for	<b>:</b>	
	UNIT $(\overline{V})$ , where	V is a double	precision	3-vector.		
	Other Library Mo	dules:				
	Execution Time (micro	seconds): 4	02.7			
		and the second of the second o				
	Input Arguments:	Precision		How Passe	eđ	Units
	Type Vector(3)	DP		$R4 \rightarrow 0$ <sup>th</sup> el	<del></del>	
	Output Results:					
	Type	Precision		How Passe	<del></del>	Units
	Vector(3)	DP		$R2 \rightarrow 0^{th} e1$	ement	
	Errors Detected:		Cauco		Fixup	
	Error #	Input vecto	<u>Cause</u>	elements=0.	-	put vector,
	<b>28</b>	Tubac vecco	T 1100 Cump			
	Comments:					
	Registers Unsafe A	cross Call:	FO,F1,F2,E	F3,F4,F5.		
	연결 등 기업을 다 살아 가게 되었다.					

Algorithm:

Loads R5 with literal 3, then drops into VV10DN code.

Secondary Entr	ry Name: VV10DN		
Function: Create in double Invoked by:	es unit vector of length precision.	n for input vector of le	ength n
$\overline{X}$ Compiler UNIT $(\overline{V})$ ,	emitted code for HAL/S where V is a double pre	construct of the form:	n, n≠3,
	brary modules:		
Execution Tim	e (microseconds): 259.	7 + 47.8n	
Input Argumen  Type  Vector (  Integer (	n) Precision DP	$\begin{array}{c} \text{How Passed} \\ \text{R4} & \rightarrow 0^{\text{th}} \text{ element} \\ \text{R5} \end{array}$	Units
Output Result Type Vector(n	Precision	$\frac{\text{How Passed}}{\text{R2}} \rightarrow 0^{\text{th}} \text{ elemen}$	<u>Units</u>
Errors Detect		Cause	Fixup
28	every element of	input vector is 0. Retur	en input vector
Comments: Registers [	Insafe Across Call: F0,	F1,F2,F3,F4,F5.	
Algorithm:	Uses loop to sum square	es of elements of input vo ot of sum. Uses loop to o by square root value and	TIATRE ERCIT

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

VV10D3	
--------	--

Secondary Entry Name:	VV9D3	<u>-</u>		
Function: Calculates	magn <b>it</b> ude of	length 3 double p	precision vecto	or.
Invoked by:				
X Compiler emitted	code for HAL/	s construct of th	ne form:	
ABVAL $(\overline{V})$ , where	V is a double	precision 3-vector	or.	
Other library mod	lules:			
Execution Time (micros	seconds): 300	.2		
Input Arguments: Type	Precision	How	Passed	Units
Vector (3)	DP	R2 → 0	th element	
Output Results: Type	Precision	<u> How</u>	Passed	Units
Scalar	DP		FO	
Errors Detected: Error #		Cause	Fix	<b>1p</b>

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: Loads R5 with literal 3, then drops into VV9DN code.

VV10D3

Secondary Entry Name	: VV9DN		
Function:			
Calculates	magnitude of length	n double precision vector.	
Invoked by:			
	d code for HAL/S com	struct of the form: sion vector of length n.	
Other library m	odules:		
Execution Time (micr	oseconds): 226.6 +	24.4n	
Input Arguments:  Type  Vector (n)  Integer (n)	Precision DP SP	R2 + Oth element.	<u>Units</u>
Output Results: Type Scalar	Precision DP	How Passed F0	<u>Units</u>
Errors Detected: Error #	Cau	se <u>Fixu</u>	<u>p</u>
Comments			

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm:

Uses loop counting down size (n), each pass squaring an element of input vector and adding to accumulated value in FO; after loop, calls DSQRT to obtain final result in FO.

			<u>vv10</u>	)S3	
	ing pagamatan di kacamatan di ka Managaran di kacamatan di kacama	AL/S-FC LIBRARY	ROUTINE DESCRIPTION		
	Source Member Name:	VV10S3	Size of Code Area:	50	Hw
	Stack Requirement:	24 Hw	Data CSECT Size:	. 2	Hw
	☐ Intrin	sic	Procedure		
	Other Library Modul	es Referenced:	SQRT, VVOSN		
-	ENTRY POINT DESCRIP	TIONS			
	Primary Entry Name:	VV10s3			
	alternative Transport and the same		ength 3 for input 3-vector	or in single	
			/S construct of the form precision 3-vector.	<b>:</b>	
	Other Library	Modules:			
	Execution Time (mid	croseconds): 2	36.4		
	Input Arguments:		36.4 How Passed		Units
		Precision SP		•	<u>Units</u>
	Input Arguments:  Type  Vector (3)	Precision	How Passed	•	<u>Units</u>
	Input Arguments:  Type  Vector (3)  Output Results:	Precision SP	$rac{ ext{How Passed}}{ ext{R4}}  ightarrow 0^{ ext{th}}  ext{ el}$	ement	<u>Units</u>
	Input Arguments:  Type  Vector (3)	Precision	How Passed	ement	-
	Input Arguments:  Type Vector (3)  Output Results:  Type Vector (3)	Precision SP Precision	$ ext{How Passed} $ $ ext{R4}  ightarrow 0^{ ext{th}}  ext{ el} $ $ ext{How Passed}$	ement <u>l</u> ement	-
	Input Arguments:  Type  Vector (3)  Output Results:  Type  Vector (3)  Errors Detected:  Error #	Precision SP  Precision SP	How Passed R4 $\rightarrow$ 0 <sup>th</sup> el How Passed R2 $\rightarrow$ 0 <sup>th</sup> el	ement i ement Fixup	Units
	Input Arguments:  Type  Vector (3)  Output Results:  Type  Vector (3)  Errors Detected:	Precision SP  Precision SP	How Passed R4 $\rightarrow$ 0 <sup>th</sup> el How Passed R2 $\rightarrow$ 0 <sup>th</sup> el	ement <u>l</u> ement	Units
	Input Arguments:  Type  Vector (3)  Output Results:  Type  Vector (3)  Errors Detected:  Error #  28	Precision SP  Precision SP	How Passed R4 $\rightarrow$ 0 <sup>th</sup> el How Passed R2 $\rightarrow$ 0 <sup>th</sup> el	ement i ement Fixup	Units

VV		

Secondary En	try Name: VV10SN		
Function:	Creates unit vector of length n in single precis	length n for input vection.	ctor of
Invoked by:			
	r emitted code for HAL/S		
UNIT (V	), where V is a single pre	ecision vector of leng	th n, n≠3.
Other 1	ibrary modules:		
Execution Ti	me (microseconds): 130.6	+ 32.8n	
Input Argume			
<u>Type</u> Vector(n Integer		$\begin{array}{c} \text{How Passed} \\ \text{R4} \rightarrow 0^{\text{th}} \text{ el} \\ \text{R5} \end{array}$	
Output Resul			Tusto.
<u>Type</u> Vector(n	Precision SP	How Passed $R2 \rightarrow 0^{th} e1$	<u>Units</u>
Errors Detec			Fixup
Error 28		Cause 11 elements = 0. Ret	- Carrier Contract Co
Comments: Register	rs Unsafe Across Call: F0	,F1,F2,F3,F4,F5.	
Algorithm:	Uses loop to sume square Calls SQRT to get square Uses loop to divide each root return value an	root of sum.	tor by square

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

VV	1	083	

Secondary Entry Name	· vv9sn		
Function: Calculates	magnitude of single	precision vector of length	n.
Invoked by:			
		nstruct of the form: dision vector of length n, n	<b>≠</b> 3.
Other library mo	odules:		
Execution Time (micro	oseconds): 118 9 +	14 On	
		基质 医双角胎的 电电	
Input Arguments:			
Type	Precision	How Passed	Units
Vector (n)	SP	$R2 \rightarrow 0$ <sup>th</sup> element	-
Integer (n)	SP		<u>-</u>
Output Results:			
Type	Precision	How Passed	Units
Scalar	SP	FO	
Errors Detected: Error #	<u>Cau</u>	<u>se</u> Fixur	

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm: Uses loop counting down size (n), each pass squaring an

element of input vector and adding to accumulated value in FO;

After loop, calls SQRT to obtain final result in FO.

Secondary Entry Name	• <u>VV9s3</u>		
Function: Calculates n	magnitude of length 3	single precision vector.	
Invoked by:			
x Compiler emitte	d code for HAL/S con	struct of the form:	
ABVAL $(\overline{V})$ , when	re V is a single prec	ision 3-vector.	
Other library m	odules:		
Execution Time (micr	coseconds): 168.3		
Input Arguments:  Type  Vector (3)	Precision SP	$\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$	Units
Output Results: Type Scalar	Precision SP	How Passed F0	<u>Units</u>
Errors Detected: Error #	<u>Cau</u>	<u>se</u> <u>Fixup</u>	

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Loads R5 with literal 3; falls into VV9SN code. Algorithm:

			VX6	D3	
АН	L/S-FC LIBRARY	ROUTINE DESCR	IPTION		
Source Member Name:	VX6D3	Size of Code	Area:	36	Hw
 Stack Requirement: _	O Hw	Data CSECT S	ize:	σ	Hw
x Intrins	ic	☐ Proced	lure		
Other Library Module	s Referenced:	None			
ENTRY POINT DESCRIPT	IONS				
Primary Entry Name:	VX6D3				
Function: Forms cre	oss product of	2 d <b>ou</b> ble preci	sion 3-vecto	ors.	
Invoked by: Compiler emitte	d code for HAL	/S construct o	f the form:		
	X and Y are do			length 3	3.
Uther Library M	odules:				
Execution Time (micr	roseconds):	137.6			
Input Arguments:					
Type Vector(3)	Precision DP		$\frac{\text{How Passed}}{\text{R2} \rightarrow \text{Oth ele}}$	ement	Units
Vector(3)	DP		R3 → Oth el	ement	••••••••••••••••••••••••••••••••••••••
Output Results:					F7 4 4
<u>Type</u> Vector(3)	Precision DP		$\frac{\text{How Passed}}{\text{Rl} \to 0^{\text{th}} \text{ el}}$	ement	<u>Units</u>
Errors Detected:					
Error #		Cause		Fixup	
Comments:					
Registers Unsafe	Across Call: 1	R1,R2,R3,R4,F0,	F1,F2,F3,F4	,F5.	
Algorithm. Direct	code no loon	s. to calculate	e cross prod	uct.	

		VX6S3	
	IAL/S-FC LIBRARY RO	UTINE DESCRIPTION	
Source Member Name:	VX6S3 s	ize of Code Area: 22	1
Stack Requirement:	O Hw D	ata CSECT Size:	0 1
☑ Intrin	sic	Procedure	
Other Library Modul	es Referenced:	NONE	
ENTRY POINT DESCRIP	TIONS		
Primary Entry Name:	VX6S3		
Function: Performs	vector cross produ	uct of two single precision	n lenath
		Jr ono ornare breersto	Length
Invoked by:			
Compiler emitt	ed code for HAL/S	construct of the form:	
X * Y where X	and Y are single r	precision 3-vectors.	
Other Library			
Other Library	Modules:		
Other Library	Modules:		
Other Library  Execution Time (mic	Modules: croseconds): 78.0		
Other Library  Execution Time (mic  Input Arguments:  Type	Modules:  roseconds): 78.0  Precision	How Passed	
Other Library  Execution Time (mic	Modules: croseconds): 78.0		
Other Library  Execution Time (mice  Input Arguments:  Type  Vector(3)  Vector (3)	Modules:  roseconds): 78.0  Precision SP	How Passed $R2 \rightarrow 0^{\text{th}} \text{ element}$	
Other Library  Execution Time (mic  Input Arguments:  Type  Vector(3)  Vector (3)	Modules:  roseconds): 78.0  Precision SP	How Passed $R2 \rightarrow 0^{\text{th}} \text{ element}$	
Other Library  Execution Time (mice  Input Arguments:  Type  Vector(3)  Vector (3)  Output Results:	Modules:  Proseconds: 78.0  Precision SP SP	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element	
Other Library  Execution Time (mic  Input Arguments:	Modules:  roseconds): 78.0  Precision SP SP Precision	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed	
Other Library  Execution Time (mic  Input Arguments:	Modules:  Proseconds): 78.0  Precision SP SP  Precision SP	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> element	xup
Other Library  Execution Time (mice  Input Arguments:	Modules:  Proseconds): 78.0  Precision SP SP  Precision SP	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> element	ixup
Other Library  Execution Time (mice  Input Arguments:	Modules:  Proseconds): 78.0  Precision SP SP  Precision SP	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> element	i <b>xup</b>
Other Library  Execution Time (mice Input Arguments:	Modules:  Proseconds): 78.0  Precision SP SP  Precision SP	How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R3 $\rightarrow$ 0 <sup>th</sup> element  How Passed  R1 $\rightarrow$ 0 <sup>th</sup> element	i <b>xup</b>

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

# 5.3.4 Character Routine Descriptions

This subsection presents those routines which manipulate character data. Routines which convert to and from character data are not included here. Such routines are found under Section 5.3.6.

	<u>CA</u>	SPV	
HAL/S-FC LIBRARY ROUTI	INE DESCRIPTION		
Source Member Name: CASPV Size	e of Code Area:	64	_ Hw
Stack Requirement: 0 Hw Data	a CSECT Size:	2	Hw
X Intrinsic	Procedure		
Other Library Modules Referenced: No	one.		
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: CASPV			
Function: Assigns a partition of a char	acter string to a t	emporary.	
Invoked by:  X Compiler emitted code for HAL/S compiler.  C\$(I TO J) where C is a character.  X Other Library Modules: CPASP  Execution Time (microseconds): (See notes)	aracter string.		
Input Arguments:  Type Precision  Character Integer (I) SP Integer (J) SP	How Passed R2 → desc R5 R6	<u>l</u> Fiptor	Units - -
Output Results:  Type Character (temporary)  Precision -	How Passed RI → desc		<u>Units</u>
Errors Detected:		Fixup	
Error # Cau  17 Indices out-of-bounds  Comments:		Set out-of index to dest characteristics.	first or

Algorithm: Several checks for possible errors are made before the transfer of characters is actually done. The index to the first character (I) is checked to be not less than 1. If it is, then set to 1. The index to the last character (J) is checked to be less than the length of the source string. If it is greater, then the index is set equal to the current length. Third, if J < I, the fixup is the NULL string. If the input actually is the NULL string, then no error is signalled. Finally, if the partition length exceeds the max length of the destination string, then the partition is truncated.

(Continued on next page)

CASPV

Execution Time:

if 
$$p = 0.43.8$$

if p > 0:

52.0 + 3.8 (if I is even)

+ 9.4k (if k is odd)

+ 13.1k (if I is even)

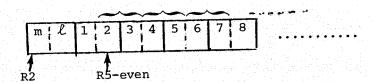
where p = minimum (J-I+1, 255)

k = ceiling (P/2)

Algorithm (Con't)

All that remains to be done is the halfword-by-halfword transfer. The character count is incremented by one before dividing by two so that the halfword count is rounded to the next highest halfword if the character count was odd.

If I (the first character index) is odd then the transfer is straightforward. If even, then there are alignment problems to work around. The odd byte of the first halfword to move must not be moved, so halfwords crossing the "natural" halfword boundary are moved instead.



Secondary Entry Name: CASP

Function: Assigns a partition of a character string to a receiver string

Invoked by:

Compiler emitted code for HAL/S construct of the form:

 $Cl = C2_{I TO J}$ , where Cl and C2 are character variables, and I and J are integers.

Other library modules:

Execution Time (microseconds): (See below)

Input Arguments:

Туре	Precision	How Passed	Units
character (C2)	- 14 - 14 - 14 - 14 - 14 - 14 - 14 - 14	R2 → descriptor	
Integer (I)	SP	R5	
Integer (J)	SP	R6	en e
Output Results:	Precision	How Passed	Units
character (C1)	<del>-</del>	Rl → descriptor	

Errors Detected:

Error #

Index out-of-bounds for input string.

Set out-of-bounds index to first or last character of string.

Comments:

Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6.

Algorithm:

Same as CASPV, except destination is a variable instead of a temporary.

Execution Time: if p = 0: 41.0 if p > 0:

where p = minimum (J-I+1, maxlength(C1))k = ceiling (P/2).

					<del></del>
			CASV		
HAL/	S-FC LIBRARY	ROUTINE I	ESCRIPTION		
Source Member Name:	CASV	Size of	Code Area:	28	Hw
Stack Requirement:	0 Hw	Data CSI	CT Size:	0 .	Hw
X Intrinsic		F	rocedure		
Other Library Modules 1	Referenced:	None.			· · · · · · · · · · · · · · · · · · ·
ENTRY POINT DESCRIPTION	NS				
Primary Entry Name:	CASV				
Invoked by:  Compiler emitted of	code for HAL	'S constru	ct of the form:		
X Other Library Modu	ıles: COUTP	, CINP	REPRODUCIE ORIGINAL P.		
X Other Library Modu Execution Time (microse if C2 ≠ null stri maxlength(Cl)), w Input Arguments:	econds) · if C	2 is null	ORIGINAL P.	age is Ro	OR
Execution Time (microse if C2 ≠ null stri maxlength(C1), w Input Arguments:	econds) · if C	2 is null	ORIGINAL P.	age is Ro	OR
Execution Time (microse if C2 ≠ null stri maxlength(C1), w Input Arguments:	econds): if C ing: 40.2 + 9 where p = min	2 is null	ORIGINAL P. string: 29.2 ng (P/2-1)) + . gth(C2), maxlen	AGE IO RO 8 (if leng gth(Cl)).	()R th(C2) >
Execution Time (microse if C2 ≠ null strimaxlength(C1)), was a substitute of the strict of the stri	econds): if C ing: 40.2 + 9 where p = min	2 is null	ORIGINAL P. string: 29.2 ng (P/2-1)) + . gth(C2), maxlen How Passed	AGE IS RO	()R th(C2) >

Registers Unsafe Across Call: R1,R2,R3,R4,R5.

Algorithm: First, the max length of the destination string is set to 255. Then, the length descriptor halfword of both the source string and the destination string are examined. The min of the max length of the destination and the current length of the source is taken as the new currlength of the destination. Next, the number of halfwords to move is found by incrementing the character count by one (in case the character cound is odd) and dividing by two. If the source is a null string, the routine exits. If the character count is odd, the last byte in the string is moved anyway since it is always ignored. The assignment is made by moving the string halfword-by-halfword to the location specified by the destination pointer.

CASV		

Secondary Entry	Name: C	AS			
Function:	Character ass	ignment, non-	partitioned.		
Invoked by:					
Compiler e	emitted code for where Cl and C	or HAL/S cons 2 are characte	truct of the form: er strings.		
X Other lib	rary modules:	CIN			
if input	<pre>(microseconds ≠ null string character str.</pre>	$: 43 + 9.4 \cdot 1$	s null string: 32.0 (ceiling(P/2-1)), when	re p = leng	<b>th</b>
Input Argument Type character	Preci	sion	How Passed R2 → descriptor		<u>Units</u>
Output Results Type character	Preci	ision	How Passed Rl → descriptor		Units
Errors Detecte Error #	: <b>d:</b>	<u>Caus</u>		Fixup	

Registers Unsafe Across Call: R1,R2,R3,R4,R5.

Algorithm: Same as CASV, except MAXLEN of destination is not set to 255, but left with original MAXLEN value.

	CATV	
DE TENENTE DE	CCRIPTION	
HAL/S-FC LIBRARY ROUTINE DE	SCRIFITON	
Source Member Name:CATV Size of C	Code Area: 76	Hw
	CT Size: 0	Hw
	ocedure	
Other Library Modules Referenced: None.		
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: CATV		
Function: Catenates two character strings an	nd stores into a temporar	ry.
어머니 보는데 아이들에 이 얼마는 하면 되었다면서 얼마나		
Invoked by:    X   Compiler emitted code for HAL/S construx   Y	ct of the form:	
Other Library Modules:		
Execution Time (microseconds): Times depend string = destination string and whether the odd character count creating an alignment pr Input Arguments:	roblem. (Continued on n	ext page).
Type Precision	How Passed	Units
Character (X)	R2 → descriptor R3 → descriptor	- 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1
Character (Y)	vo - describeor	
Output Results:		Units
Type <u>Frecision</u>	How Passed	UNITES
Character(temporary) -	Rl → descriptor	The second
Errors Detected: Error # Cause	Fixup	
none		

Comments: Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6,R7,F0,F1.

Algorithm: The lengths of the source strings are checked against the destination string for legal values. The second source string may be truncated if its length + that of the first source string exceed the length of the destination. If the first source string and the destination string are the same string (found by comparing addresses), then only the second source string is moved. After checking these things, the routine needs only to actually move the strings. The first is a straight halfword-by-halfword move. If its length is odd, then there is the alignment problem to contend with. The second string is moved starting where the first one left off. See description of CASPV for what is done when the first source string has an odd character count.

CATV

Execution Time (microseconds): (Cont'd.)

if X is null string and Y is null: 52.2 µsec.

if X and Y are not both null: XTIME + YTIME.

XTIME: if X is null string: 24.0

if  $X \neq \text{null string: } 29.8 + 9.4 \text{ (ceiling (P/2))}$ 

where p = length(X).

YTIME: if Y is null string: 27.8

if Y is ≠ null string:

 $52.1 + 14.1 \cdot (ceiling(Q/2-1))$  if P is odd + 6.0 (if P+Q is odd)

 $32.3 + 9.4 \cdot (ceiling (Q/2))$  if P is even

where Q = minimum (length(Y), 255-P).

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

CATY	

Secondary Entry Name:	CAT	<b></b>			
Function: Catenates to	wo character s	strings and s	tores into a	third string	ſ <b>.</b>
Invoked by:					
X Compiler emitted Not used yet.	code for HAL/	S construct o	of the form:		
Other library mod	ules:				
Execution Time (micros	econds): <sub>San</sub>	me as CATV -	2.5		
Input Arguments: Type Character Character	Precision		How Passed R2 → descript R3 → descript		Units -
Output Resulis:  Type  Character	Precision -		How Passed Rl → descript	or	<u>Units</u>
Errors Detected: Error #		Cause		Fixup	

Comments:

Registers Unsafe Across Call: Rl,R2,R3,R4,R5,R6,R7,F0,F1.

Algorithm:

Same as CATV, except string is moved to real data area, not

to a temporary.

 	CINDEX	
HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
CINDIA	Size of Code Area: 52 Ho	ω,
Source Member Name:	o o	
Stack Requirement: 18 Hw	Data CSECT Size: Ho	<i>N</i>
☐ Intrinsic	Y Procedure	
Other Library Modules Referenced:	GTBYTE	
ENTRY POINT DESCRIPTIONS		•
Primary Entry Name: CINDEX		•
Function: Performs HAL/S INDEX fur string within another.	action: finds occurrance of one charact	er
Invoked by:  [X] Compiler emitted code for HAL  INDEX(A,B), where A and B ar  within A	re character strings; B is searched for	
Other Library Modules:		
Execution Time (microseconds): (Se	ee next page).	
Tyre Precision		Units
character(A) - character(B) -	R2 → descriptor R4 → descriptor	
Output Results:	How Passed	Units
Type Precision Integer SP	R5	
Errors Detected:		
Error #	<u>Cause</u> <u>Fixup</u>	
Comments:		
Registers Unsafe Across Call		
2) Set pointer to fir 3) If size of B excee 4) Loop on size of B, at current A point 5) Comparison loop in	s null, return zero. st character of A. ds size of A, beyond A pointer, return comparing elements of A and B beginninger; on non-equality go to step 6. 4 succeeded, return current A pointer or by one byte, go to step 3.	ng

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

#### CINDEX

Execution Time (microseconds):

if A is null: 32.8

if B is null: 38.0

if length(B) > length(A): 44.8

if result = 0:

time = 38.0 + 
$$\sum_{I=1}^{JA}$$
 ( $\sum_{J=1}^{KA_I}$  (15.4 +  $KB_J$  +  $KB_{J+I-1}$ ) + 16.4 +  $KB_I$ )

where JA = 2 (length(C1) - length(C2))+1

 $KA_{I}$  = # of compares required to determine that Cl\$(length(C2) at I)  $\gamma$  = C2.

$$KB_X = \begin{cases} 14.4 \text{ if } X \text{ is even} \\ 15.6 \text{ if } X \text{ is odd.} \end{cases}$$

if result = 0:

result 
$$KA_{I}$$
  
time = 29.6 +  $\Sigma$  ( $\Sigma$  (15.4 +  $KB_{J}$  +  $KB_{J+I-1}$ ) + 16.4 +  $KB_{I}$ )  
 $I=1$   $J=1$ 

- KBresult

 $KB_{\chi}$  is as above.

	CLJSTV	
HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
Source Member Name: CLJSTV	Size of Code Area: 40	Hw
Stack Requirement: 18 Hw	Data CSECT Size: 2	Hw
☐ Intrinsic	Procedure	
Other Library Modules Referenced: _	GTBYTE, STBYTE	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: CLJSTV		
Function: Left justifies a charact 1) padding on the right 2) truncating on the ri	with blanks if too short;	р <b>у</b>
Invoked by:  X Compiler emitted code for HAL/	'S construct of the form:	
LJUST(A,B), where A is a chara B is an inte	acter string, and	
Other Library Modules:		
Execution Time (microseconds): (8	See next page).	
Input Arguments:	How Passed	Units
Type Precision		<u> </u>
character(A) -	R4 → descriptor	
integer (B) SP	<b>R5</b>	
Output Results:		Units
Type Precision	How Passed	Ulites
character(temporary) -	R2 → descriptor	
Errors Detected:	Cause Fixup	
Error #  18 Input string leng reques or B	th greater than Truncate inputed size. to specified	
Comments: Registers Unsafe Across Call:	FO,F1.	
if same, moves input	th to 255 and retains smaller as tring length; on right to length L and moves t string unchanged to output; t with blanks and moves to output	o output;

### CLJSTV

Execution Time:

34.0 + 2.8 (if B < 255)

+ 2.0 (if n > 0)

+ 40.8n

+ 1.6 (if n is odd)

 $+ 0.4 \text{ (if } m \leq 0)$ 

+ 1.0 (if m is odd and n is even)

- 1.0 (if m is odd and n is odd)

+ 23.8 m

where n = length(A)

m = maximum(B-n, 0)

		_C:	PAS	
HAL/S-F	C LIBRARY RO	UTINE DESCRIPTION		
Source Member Name:	PAS S	ize of Code Area: _	80	Hw
Stack Requirement:	_20 Hw D	oata CSECT Size:	2	Hw
☐ Intrinsic		Procedure		
Other Library Modules Re	ferenced:	GTBYTE, STBYTE		
ENTRY POINT DESCRIPTIONS				
Primary Entry Name:	CPAS			
Function: Assigns a ch	naracter stri	ng to a partition of	another str	ing.
Invoked by:	5- 6 UNT /C	construct of the fo	rm:	
Compiler emitted co	de IOI HAL/S	CONSCIUCE OF CHE TO		
$C2_{T,TO,T} = C1;$	Cl and C2 ar	e character strings.		
<u> </u>				
X Other Library Modul	es: CPASP, C			
X Other Library Modul	es: CPASP, C			
		CINP		
X Other Library Modul Execution Time (microsec		CINP		
Execution Time (microsec		CINP		
Execution Time (microsecon Input Arguments:	onds): (See	CINP next page).		Unit
Execution Time (microsecon Input Arguments:  Type  Execution Time (microsecon Input Arguments:	onds): (See	CINP	:e <b>d</b>	<u>Uni</u>
Execution Time (microsecon Input Arguments:  Type Per Character string (source)	onds): (See	next page).  How Pass $R4^{2} \rightarrow desc$ $R5$	:e <b>d</b>	<u>Unit</u>
Execution Time (microsecon Input Arguments:  Type  P	onds): (See	next page).  How Pass $R4^- \rightarrow desc$	:e <b>d</b>	Uni
Execution Time (microsecon Input Arguments:  Type character string(source integer(I)	onds): (See	next page).  How Pass R4 + desc R5 R6	sed criptor	
Execution Time (microsecon Input Arguments:  Type Property Propert	onds): (See	next page).  How Pass R4 + desc R5 R6 How Pass	sed riptor sed	
Execution Time (microsecon)  Input Arguments:  Type P  character string(source integer (I) integer (J))  Output Results:	onds): (See	next page).  How Pass R4 + desc R5 R6 How Pass	sed criptor	
Execution Time (microsecon)  Input Arguments:  Type Properties of the properties of	onds): (See	next page).  How Pass R4 + desc R5 R6 How Pass	sed riptor sed	
Execution Time (microsecon Time Type Property Pr	recision (C1) Precision C2) -	next page).  How Pass R4 + desc R5 R6 How Pass	sed escriptor	Uni
Input Arguments:  Type P character string(source integer (I) integer (J) Output Results:  Type character (destination)  Errors Detected: Error #	onds): (See	next page).  How Pass R4 → desc R5 R6  How Pass R2 → dc	sed escriptor Fixup Set out-of-	Uni Uni -
Execution Time (microsecon)  Input Arguments:  Type Properties of the properties of	onds): (See	How Pass R4 → desc R5 R6  How Pass R2 → de	sed escriptor	Uni Uni -
Input Arguments:  Type P  character string(source integer (I) integer (J)  Output Results:  Type character (destination)  Errors Detected: Error #	onds): (See	next page).  How Pass R4 + desc R5 R6  How Pass R2 + de	sed escriptor Fixup Set out-of-	Uni Uni - bounds in last cha
Input Arguments:  Type Percharacter string (source integer (I) integer (J)  Output Results:  Type character (destination)  Errors Detected:  Error #  17 Index  Comments:	onds): (See Precision (C1) - Precision (C2) -  out-of-bound or J < I-	next page).  How Pass R4 + desc R5 R6  How Pass R2 + de	sed escriptor  Fixup Set out-of- to first or	Unit Unit - bounds ir last cha
Input Arguments:  Type Percharacter string (source integer (I) integer (J)  Output Results:  Type character (destination)  Errors Detected:  Error #  17 Index  Comments:	onds): (See Precision (C1) - Precision (C2) -  out-of-bound or J < I-	next page).  How Pass R4 + desc R5 R6  How Pass R2 + de	sed escriptor  Fixup Set out-of- to first or	last cha

partition is longer, then pad with blanks. The character count is determined and the string is moved byte-by-byte with the GTBYTE and STBYTE routines.



**CPAS** 

Execution Time (microseconds):

34.2 + KA + 
$$\sum_{k=1}^{LHP}$$
 (5.6 + KC<sub>LOUT+K</sub>) + KD  
+  $\sum_{k=1}^{NCHAR}$  (7.6 + KC<sub>I+K-1</sub> + KF<sub>k</sub>) + KE  
+  $\sum_{k=1}^{RHP}$  (5.6 + KC<sub>I+LIN+K-1</sub>) + KG

where

LOUT = length(C2) before assignment

LIN = length(Cl)

$$KA = \begin{cases} 25.4 & \text{if } J \le LOUT \\ 34.0 & \text{if } J > LOUT \end{cases}$$

LPART = J-I+1 (length of partition)

LHP = I-LOUT-1

$$KC_X = \begin{cases} 19.2 \text{ if } X \text{ is odd} \\ 17.2 \text{ if } X \text{ is even} \end{cases}$$

$$KD = \begin{cases} 4.0 & \text{if LHP} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

NCHAR = MINIMUM (LPART, LIN)

$$KE = \begin{cases} .8 & \text{if NCHAR} = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$KF_X = \begin{cases} 15.6 \text{ if } X \text{ is odd} \\ 14.4 \text{ if } X \text{ is even} \end{cases}$$

RHP = LPART-LIN

Note that in summations, if start\_index > end\_index then summation goes to 0.

	CPASP	
HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
Source Member Name:CPASP	Size of Code Area: 16	_ Hw
Stack Requirement: 146 Hw	Data CSECT Size: 0	Hw
☐ Intrinsic	X Procedure	
Other Library Modules Referenced:	CASPV, CPAS	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: CPASP		
Function: Assigns a partition of another character strip	f a character string into a parti <b>tio</b> ng.	<b>on</b>
Invoked by:  X Compiler emitted code for HAL	/S construct of the form:	
C1 = C2; $C1$ and	d C2 are character strings, and ,L are integers.	
Other Library Modules:		
Execution Time (microseconds): 4:  Input Arguments:	2.8 + time for CASPV and CPAS.	
Type Precision	How Passed	Units
character(source)C2 - integer(K) SP integer(L) integer(I)(SP SP)	R4 → descriptor R5 R6 R7	
integer I   J) (SP   SP) Output Results:  Type Precision character (destination) Cl	How Passed RI → descriptor	Units
Errors Detected:		
Error #	Cause Fixup	
17 Subscript of char	bounds. Set out-of-b to first or character of	last
Comments:	string.	. associate
Registers Unsafe Across Call:	FO, F1.	
Algorithm: The input partition is	s put into a VAC by the CASPV routination string and pointers are set unoves the contents of the VAC into the	ıp

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

		<u>C</u>	PR
<b>H</b>	AL/S-FC LIBRARY	ROUTINE DESCRIPTION	•
Source Member Name:	CPR	Size of Code Area:	46 Hw
Stack Requirement:	O Hw	Data CSECT Size:	O Hw
☑ Intrins	sic	Procedure	
Other Library Module	es Referenced:Nor	ne.	
ENTRY POINT DESCRIPT	TIONS		
Primary Entry Name:	CPR		
Function: Compares condition		rings for '=' or '¬='	and sets
		S construct of the form C2 are character string	
X Other Library M	odules: CPRA		
Execution Time (micr	coseconds): (se	ee next page).	
'선생님 시간 [			
Input Arguments:  Type Character (C1) Character (C2)	Precision	How Passed R2 → desc R3 → desc	riptor
Output Results:			
Type equal/not equal	Precision	How Passed condition	
Errors Detected.			

Comments: In order to not change the condition code after the comparisons and before exiting, instructions that change the c.c. are replaced by those that do not change it. For example, LH replaced by IHL and SLL. (Cont'd. below) Algorithm: See CPRC entry.

Cause

Fixup

Comments: (Cont'd.)

Error #

Registers Unsafe Across Call: R2, R3, R4, R5, R6.

CPR

Execution Time (microseconds):

If C1 and C2 do not halfword compare and  $K \ge 2$ :

setup + 11.6J + 12.9

If K is even or K = 0 and C1 and C2 halfword compare 1 up till the  $K^{\mbox{th}}$  character:

setup + 11.6n + 20.1

If K is odd and Cl = C2 up till the K<sup>th</sup> character:

setup + 11.6n + 29.9

If K is odd and only the last characters compared differ

setup + 11.6n + 20.3

where:

K = minimum(length(C1), length(C2))

setup = 23 + .4 (if length(C2) < length(C1))

J = number of matching halfword compares

n = floor (K/2)



Function: Compares two character strings for collating sequence and sets condition code.

Invoked by:

X Compiler emitted code for HAL/S construct of the form:
If C1 < C2..., or any other relational operator, except '=', or ' ¬='.

Other library modules:

Execution Time (microseconds): Same as CPR

Input Arguments: Type	Precision	How Passed	Units
character (Cl) character (C2)		R2 - descripter R3 - descripter	
Output Results: Type	Precision	How Passed	Units
relation		condition code	<u> -</u>
Errors Detected: Error #	Cause	Fixup	

Comments:

See CPR

Registers Unsafe Across Call: R2,R3,R4,R5,R6.

Algorithm: Find the smaller of the lengths of the two strings to be compared. Compare this many characters halfword-by-halfword, and compare the upper bytes of the last halfwords separately if the character count is odd. If any of these comparisons are unequal, then return the resultant condition code. If all are equal, then compare the lengths of the two strings, and return the resultant code.

			CPRA	<del></del>
•	HAL/S-FC LIBE	ARY ROUTINE DESC	CRIPTION	
	Source Member Name: CPRA	Size of Cod	le Area: 29	Hw
	Stack Requirement: 22	Hw Data CSECT	Size: 0	Hw
	☐ Intrinsic	X Proce	dure	
	Other Library Modules Reference	ed: CPR		
	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: CPRA			
	Function: Compares arrays of	character string	<b>S.</b>	
	Invoked by:  [X] Compiler emitted code for	HAL/S construct	of the form:	
	<pre>IF S1 = S2, where S1 ar    is a lengt</pre>	nd S2 are structu h n array of cha		odes
	Other Library Modules:			
	Execution Time (microseconds):	NCMP $23.2 + \Sigma$ (18)	3.2 + CPRTIME <sub>k</sub> ) - 14	.2 (if array
	(Cont'd. below).	k=1.		re not equal
	Input Arguments:			Tinita
	Type Precisi character array	<u>on</u> _	How Passed R2 → Oth element	Units
	character array		$R3 \rightarrow 0^{th}$ element	_
	<pre>integer(# Hw in each string) { integer(n)</pre>		R6 R7	
	Output Results:  Type Precisi		How Passed	Units
	equal/not equal -		condition code	
	Errors Detected:			
	Error #	Cause	<u>Fixu</u>	<u>p</u>
	Comments:			
	Registers Unsafe Across	Call: None.		

Algorithm: Pointers to character strings within the array are set, then CPR routine called. If all pairs of strings within the array are equal, result of CPRA is "equal", otherwise the result is "not equal".

Execution Time (Cont'd.)

where NCMP = number of elements in arrays if arrays are equal, index of first non-matching character strings in arrays if arrays not equal.

CPRTIME = time in CPR for S1.C\$(X:) and S2.C\$(X:) where C is the node for the array of character strings.

	CRJSTV	
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION	
	and the control of th	Hw
	2	Hw .
	Stack Requirement:	
,		
	Other Library Modules Referenced: GTBYTE, STBYTE	
	ENTRY POINT DESCRIPTIONS	
	Primary Entry Name: CRJSTV	•
	Function: Right-adjusts a character string to a specified length by:  1) padding on left with blanks if too short; 2) truncating on left if too long.  Invoked by:  X Compiler emitted code for HAL/S construct of the form: RJUST(A,B), where A is a character string, and B is an integer.	
	Other Library Modules:	
	Execution Time (microseconds): (See next page).	
	Input Arguments.  Type Precision How Passed R4 $\rightarrow$ descriptor integer (B) SP R5	Units
	Output Results:  Type Precision How Passed  Character (temporary) - R2 $\rightarrow$ descriptor	Units
	Errors Detected:  Error #  18 Input string length greater than input size or B < 0.  Fixup  Truncate input on left to provide the provided in	
	Comments:	
	Registers Unsafe Across Call: F0,Fl.	
	Algorithm: Compares requested length to 255 and retain smaller as L;  Compares current length with L:  if greater, truncates on left and moves to input;  if same, moves string to output;  if less, pads on left and moves input string to output.	

#### CRJSTV

Execution Time (microseconds):

$$\begin{array}{cccc}
 & \text{NBLANK} \\
 & 34.8 + \text{KA} + \sum & (6.8 + \text{KB}_{k}) + \text{KC} \\
 & & \text{k=1}
\end{array}$$

NCHAR  
+ 
$$\Sigma$$
 (5.6 + KD<sub>k</sub> + KB<sub>NBLANK+K</sub>) + KE  
k=1

where:

$$KA = \begin{cases} 0 \text{ if } B \ge 255 \\ 2.8 \text{ if } B \le 255 \end{cases}$$

NBLANK = 
$$\begin{cases} B-length(A) & \text{if } B > length(A) \\ 0 & \text{otherwise} \end{cases}$$

$$KB_{X} = \begin{cases} 19.2 \text{ if } X \text{ is odd} \\ 17.2 \text{ if } X \text{ is even} \end{cases}$$

$$KC = \begin{cases} 1.2 & \text{if NBLANK} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$KD_{X} = \begin{cases} 15.6 \text{ if } X \text{ is odd} \\ 14.4 \text{ if } X \text{ is even} \end{cases}$$

$$KE = \begin{cases} .4 & \text{if NCHAR} = 0 \\ 0 & \text{otherwise} \end{cases}$$

		CTF	RIMV	
	HAL/S-FC LIBRARY ROUT	INE DESCRIPTION		
	Source Member Name: CTRIMV Siz	e of Code Area:	94	Hw
	Stack Requirement: 18 Hw Dat	a CSECT Size:	0.	Hw
	☐ Intrinsic	Procedure		
	Other Library Modules Referenced:	GTBYTE, STBYTE	·	
·	ENTRY POINT DESCRIPTIONS		<del></del>	
	Primary Entry Name: CTRIMV			
	Function: Implements HAL/S TRIM functi trailing blanks from a charac		and	
	Invoked by:  X Compiler emitted code for HAL/S con	nstruct of the form	<b>:</b>	
	TRIM(C), where C is a character s	tring.		
	Other Library Modules:			
	Execution Time (microseconds): (See	next page).		
	Input Arguments:			
	Type Precision	How Passed	•	Units
	character (C)	R4 → descri	ptor	<del>-</del>
	Output Results:  Type Precision  character (temporary) -	How Passed R2 → descri		Units
	Errors Detected: Error # Caus	<u>se</u>	Fixup	

Comments:

Registers Unsafe Across Call: FO,Fl.

Algorithm: Because there are no character or byte compare instructions on the AP-101, the routine first tests length of string. If odd, it sets R7 to 1. "Length" is shifted right 1, resulting in length in # of halfwords (-1 if odd). Compares first halfword with bb, continues comparing consecutive halfwords of string with bb, until a halfword that is not equal to bb is found. Then tests this halfword to see if first byte is b. Adds length of string in halfwords to pointer to string, resulting in a pointer to end of string. Compares last halfword of string with bb. If equal, then moves pointer back a halfword and again compares. When a halfword not equal to bb is found, the halfword is tested to see if it is Cb or CC (where C stands for any character). Length of string is appropriately adjusted, and routine branches to a character move loop.

#### CTRIMV

Execution Time (microseconds):

If length(C) = 0: 30.4

If length(C) = 1 and C is a blank: 64.0

If length(C) = 1 and C is not a blank: 102.8

If length(C) > 1 and all blank

44.6 + KA + 13.2KB

where:

KB = floor (length(C)/2)

If length(C) > 1 and not all blank

$$60.4 + KA + 13.2 \cdot KB + KC + KD(11.6 \cdot KE + 13.6 + KF)$$

NCHAR

$$+ \sum_{k=1}^{\infty} (39.2 + KG_{KH+k-1} + KI_{k})$$

where:

KA = 0 if length(C) is even

19.4 if length(C) is odd and C\$(#) is blank

18.4 if length(C) is odd and  $C$(\#) \neg = blank$  and  $C$(\#) \neg = null$ 

22.4 if length(C) is odd and  $C$(\#) \neg = blank$  and C\$(#) = null

KB = # halfwords = blank | blank at the beginning of C

KC = 0 if index of first non\_blank character is odd and this character = null

4.8 if index of first non blank character is odd and this character = null

9.0 if index of first non blank character is even.

KD = 0 if length(C) is odd and (C\$(#) ¬ = blank
1 otherwise.

KE = # halfwords = blank | blank at the end of C

(Continued on next page)

#### CTRIMV

### Execution Time (Continued):

3.6 if index of last untrimmed character is even and this character = null

4.4 if index of last untrimmed character is odd.

NCHAR = length of result.

 $KG_X = 0$  if X is even 2.0 if X is odd

KH = index of first non blank character

 $KI_X = 0$  if X is odd 1.2 if X is even

# 5.3.5 Array Function Routine Descriptions

This subsection presents those routines which are classed as "ARRAY FUNCTIONS" by the HAL/S Language Specification. These are routines which operate upon arrayed arguments and produce a single element result.

REPRODUCEMENT OF THE CORRECTION OF THE

		. · · · · ·	DMAX	_
	HAL/S-FC LIBRAN	RY ROUTINE DESCRIPTION		
;	Source Member Name:DMAX	_ Size of Code Area:	10	Hw
	Stack Requirement: 0 H	w Data CSECT Size:	0	Hw
	X Intrinsic	Procedure		
	Other Library Modules Referenced	: None		
	ENTRY POINT DESCRIPTIONS			
;	Primary Entry Name: DMAX			
	Function: Finds maximum value	in a double precision	scalar array.	
	Invoked by:			
	Compiler emitted code for H	AL/S construct of the i	form:	
	MAX( <dp array="">)</dp>	AL/S construct of the i	form;	
	MAX( <dp array="">)</dp>	AL/S construct of the i	form:	
		AL/S construct of the i	form;	
	MAX( <dp array="">)</dp>	7.6L + 14.6m + 11.4, w	here L = # of	or erem
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1 CURRMAX changes; M = # of times in array)-1. Input Arguments:  Type Precision</dp>	7.6L + 14.6m + 11.4, w CURRMAX does not change How Pa:	here L = # of e, L+M = (# o ssed	times f elen
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1 CURRMAX changes; M = # of times in array)-1. Input Arguments:</dp>	7.6L + 14.6m + 11.4, w CURRMAX does not change How Pa:	here L = # of e, L+M = (# o ssed element	or eren
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1  CURRMAX changes; M = # of times in array)-1.  Input Arguments:  Type Precision  Scalar array DP  Integer(size) SP</dp>	7.6L + 14.6m + 11.4, where $0$ CURRMAX does not change $0$ How Pass $0$ R2 $\rightarrow$ 0th	here L = # of e, L+M = (# o ssed element	or eren
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1 CURRMAX changes; M = # of times in array)-1. Input Arguments:  Type Precision Scalar array DP</dp>	7.6L + 14.6m + 11.4, we curred to change $\frac{\text{How Pas}}{\text{R2} \to \text{Oth}}$	here L = # of e, L+M = (# o ssed element	or eren
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1 CURRMAX changes; M = # of times in array)-1. Input Arguments:  Type Precision Scalar array DP Integer(size) SP  Output Results:</dp>	7.6L + 14.6m + 11.4, we curred to change $\frac{\text{How Pas}}{\text{R2} \to \text{Oth}}$	here L = # of e, L+M = (# o ssed element ssed	í
	MAX( <dp array="">)  Other Library Modules:  Execution Time (microseconds): 1 CURRMAX changes; M = # of times in array)-1. Input Arguments:  Type Precision Scalar array DP Integer(size) SP  Output Results: Type Precision</dp>	7.6L + 14.6m + 11.4, which was not change $\frac{\text{How Pass}}{\text{R2} \to \text{0th}}$	here L = # of e, L+M = (# o ssed element ssed	í

A loop is set up to compare each element of the array to the Algorithm: current max. Initially, the first element is CURRMAX, Each subsequent element of greater value replaces the former CURRMAX. The counter is decremented after each comparison. The value of CURRMAX when the counter is zero is the max of the array and is passed back to the calling program.

			-	DMIN	
на	L/S-FC LIBRARY	ROUTINE DESCRI	PTION		
Source Member Name:	DMIN	Size of Code	Area:	10	Hw
Stack Requirement: _	0Hw	Data CSECT Si	Lze:		_ Hw
X Intrins	ic	☐ Procedu	ıre		
Other Library Modules	s Referenced:	None.			
ENTRY POINT DESCRIPT	IONS				
Primary Entry Name:	DMIN				
Function: Finds mini	mum value of a	a double precis	ion scalar a	array.	
Invoked by:  X Compiler emitted	d code for HAI	/S construct o	f the form:		
MIN( <dp scala<="" td=""><td>r array&gt;)</td><td></td><td></td><td></td><td></td></dp>	r array>)				
Other Library M	odules:				
Execution Time (micr CURRMIN changes; M = in array)-1.	oseconds): 17 # of times CUI	$.6L \div 14.6m + 1$ RRMIN does not	1.4, where I change; L+M	= # of t: = (# of e	Lmes Lements
Input Arguments: Type	Precision		How Passed		Units
Scalar array	DP	R2	→ O <sup>th</sup> elemen	nt	- -
Integer(size)	SP		R5		
Output Results:			Hott Dagged		Units
<u>Type</u> Scalar	Precision DP		How Passed F0		OHLC
Errors Detected: <pre>Error #</pre>		Cause		Fixup	
Comments: Registers	s Unsafe Acros	s Call: R2, R4	, R5, F0, F		
· Algorithm: Similar current minimum and	d is changed w	ions, except re hen an element	gister cont in the arra	ains the y has a	

REPRODUCIBILITY OF THE ORIGINAL PAGE TO POOR

			DPROD	
HAL	/S-FC LIBRARY R	ROUTINE DESCRIPTION		
Source Member Name: _	DPROD	Size of Code Area:	14	Hw
Stack Requirement:	O Hw	Data CSECT Size:	0	Hw
X Intrinsi	C	Procedure		
Other Library Modules	Referenced:	None.		
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name: _	DPROD			
Function: Calculates scalar array		the elements of a	double precis	ion
Invoked by: KX Compiler emitted	ando for HAT./	dtrust of the	form:	
PROD ( <dp scala<="" td=""><td>r array&gt;)</td><td>s construct or the</td><td></td><td></td></dp>	r array>)	s construct or the		
PROD( <dp scala<="" td=""><td>ar array&gt;)</td><td></td><td></td><td></td></dp>	ar array>)			
PROD ( <dp (micron="#" elements="" execution="" index="" into="" library="" line<="" mother="" of="" other="" scalar)="" td="" the="" time=""><td>odules:</td><td>5n + 6.2 if product</td><td>is not zero,</td><td></td></dp>	odules:	5n + 6.2 if product	is not zero,	
PROD ( <dp (micr<="" execution="" library="" m="" other="" scala="" td="" time=""><td>odules:</td><td>on + 6.2 if product 20.6m + 2.6 if pr ion of the array of How Pa</td><td>is not zero, oduce is zero the first zer</td><td>o element</td></dp>	odules:	on + 6.2 if product 20.6m + 2.6 if pr ion of the array of How Pa	is not zero, oduce is zero the first zer	o element
PROD ( <dp (micron="#" arguments:<="" elements="" execution="" index="" input="" into="" library="" line="" mother="" of="" other="" scalar)="" td="" the="" time=""><td>odules: oseconds): 20.6 In the array. car representati</td><td>on + 6.2 if product 20.6m + 2.6 if pr ion of the array of How Pa</td><td>is not zero, oduce is zero the first zer ssed</td><td>o element</td></dp>	odules: oseconds): 20.6 In the array. car representati	on + 6.2 if product 20.6m + 2.6 if pr ion of the array of How Pa	is not zero, oduce is zero the first zer ssed	o element
PROD ( <dp (size)="" arguments:="" array="" index="" input="" integer="" into="" library="" line="" mother="" other="" output="" results:<="" scalar="" scalar)="" td="" the="" type=""><td>odules:  oseconds): 20.6  n the array. ear representation  DP  SP</td><td><math>6n + 6.2</math> if product <math>20.6m + 2.6</math> if prion of the array of <math display="block">\frac{\text{How Pa}}{\text{R2} \rightarrow 0^{\text{th}}}</math></td><td>is not zero, oduce is zero the first zer ssed element</td><td></td></dp>	odules:  oseconds): 20.6  n the array. ear representation  DP  SP	$6n + 6.2$ if product $20.6m + 2.6$ if prion of the array of $\frac{\text{How Pa}}{\text{R2} \rightarrow 0^{\text{th}}}$	is not zero, oduce is zero the first zer ssed element	
PROD ( <dp (size)<="" arguments:="" array="" index="" input="" integer="" into="" library="" line="" mother="" other="" scalar="" scalar)="" td="" the="" type=""><td>odules:  oseconds): 20.6  n the array. ear representation  DP</td><td><math>6n + 6.2</math> if product <math>20.6m + 2.6</math> if prion of the array of <math display="block">\frac{\text{How Pa}}{\text{R2} \rightarrow 0^{\text{th}}}</math></td><td>is not zero, oduce is zero the first zer ssed</td><td>o element <u>Unit</u> -</td></dp>	odules:  oseconds): 20.6  n the array. ear representation  DP	$6n + 6.2$ if product $20.6m + 2.6$ if prion of the array of $\frac{\text{How Pa}}{\text{R2} \rightarrow 0^{\text{th}}}$	is not zero, oduce is zero the first zer ssed	o element <u>Unit</u> -

Comments: Registers Unsafe Across Call: R2, R4, R5, F0, F1.

Algorithm: Similar to the algorithm for the SUM functions. An accumulator is initialized to one. The value in the accumulator is multiplied by each element of the array; the result of each multiplication is saved in the accumulator. After each multiplication, the result is checked for a zero product. If the product is ever zero, the routine exits and returns zero.

			DSUM	
на	L/S-FC LIBRARY R	OUTINE DESCRIPTI	ON	
Source Member Name:	DSUM	Size of Code Are	ea: 6	Hw
Stack Requirement: _	O Hw	Data CSECT Size:	0	Hw
[X] Intrins	ic	<pre>Procedure</pre>		
Other Library Module	s Referenced: No	one.		
 ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	DSUM	<u> </u>		
Function: Calcul scalar		the elements of	a double precis	ion
Invoked by:			ho form:	
Compiler emitte SUM( <dp scal<="" th=""><th>ar array&gt;)</th><th></th><th></th><th>ts in the</th></dp>	ar array>)			ts in the
SUM( <b><dp< b=""> scal</dp<></b>	ar array>)		n = # of elemen	ts in the
SUM( <dp (micr<="" execution="" library="" m="" other="" scal="" td="" time=""><td>ar array&gt;)</td><td>+ ll.6n, where array.</td><td>n = # of elemen</td><td></td></dp>	ar array>)	+ ll.6n, where array.	n = # of elemen	
SUM( <dp scal<="" td=""><td>ar array&gt;)</td><td>+ 11.6n, where array.</td><td>n = # of elemen</td><td></td></dp>	ar array>)	+ 11.6n, where array.	n = # of elemen	
SUM ( <dp (micr="" arguments:<="" execution="" input="" library="" m="" other="" scal="" td="" time=""><td>ar array&gt;) dodules: coseconds): 7.2</td><td>+ 11.6n, where array.</td><td>n = # of elemen Passed Oth element</td><td></td></dp>	ar array>) dodules: coseconds): 7.2	+ 11.6n, where array.	n = # of elemen Passed Oth element	
SUM( <dp (micr="" arguments:="" execution="" input="" library="" m="" other="" scal="" td="" time="" type<=""><td>ar array&gt;)  Modules:  Coseconds): 7.2  Precision</td><td>+ 11.6n, where array.</td><td>n = # of elemen</td><td></td></dp>	ar array>)  Modules:  Coseconds): 7.2  Precision	+ 11.6n, where array.	n = # of elemen	
SUM( <dp (micr="" arguments:="" array<="" execution="" input="" library="" m="" other="" scal="" scalar="" td="" time="" type=""><td>ar array&gt;) Modules: Coseconds): 7.2  Precision DP SP</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of element  Passed  Oth element  R5</td><td><u>Uni</u></td></dp>	ar array>) Modules: Coseconds): 7.2  Precision DP SP	+ 11.6n, where array.  How R2 →	n = # of element  Passed  Oth element  R5	<u>Uni</u>
SUM( <dp (micr="" arguments:="" array="" execution="" input="" integer(size)<="" library="" m="" other="" scal="" scalar="" td="" time="" type=""><td>ar array&gt;) Modules: Coseconds): 7.2  Precision DP</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of elemen Passed Oth element R5</td><td></td></dp>	ar array>) Modules: Coseconds): 7.2  Precision DP	+ 11.6n, where array.  How R2 →	n = # of elemen Passed Oth element R5	
SUM( <dp (micr="" arguments:="" array="" execution="" input="" integer(size)="" library="" m="" other="" output="" results:<="" scal="" scalar="" td="" time="" type=""><td>ar array&gt;) Modules: Coseconds): 7.2  Precision DP SP</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of element  Passed  Oth element  R5</td><td><u>Uni</u></td></dp>	ar array>) Modules: Coseconds): 7.2  Precision DP SP	+ 11.6n, where array.  How R2 →	n = # of element  Passed  Oth element  R5	<u>Uni</u>
SUM( <dp (micr="" arguments:="" array="" execution="" input="" integer(size)="" library="" m="" other="" output="" results:="" scal="" scalar="" td="" time="" type="" type<=""><td>ar array&gt;) dodules: coseconds): 7.2  Precision DP SP  Precision</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of element  Passed Oth element R5 Passed F0</td><td>Uni Uni</td></dp>	ar array>) dodules: coseconds): 7.2  Precision DP SP  Precision	+ 11.6n, where array.  How R2 →	n = # of element  Passed Oth element R5 Passed F0	Uni Uni
SUM( <dp (micr="" arguments:<="" execution="" input="" library="" m="" other="" scal="" td="" time=""><td>ar array&gt;) dodules: coseconds): 7.2  Precision DP SP  Precision</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of elemen Passed Oth element R5</td><td><u>Uni</u></td></dp>	ar array>) dodules: coseconds): 7.2  Precision DP SP  Precision	+ 11.6n, where array.  How R2 →	n = # of elemen Passed Oth element R5	<u>Uni</u>
SUM( <dp (micr="" arguments:<="" execution="" input="" library="" m="" other="" scal="" td="" time=""><td>ar array&gt;) dodules: coseconds): 7.2  Precision DP SP  Precision</td><td>+ 11.6n, where array.  How R2 →</td><td>n = # of element  Passed Oth element R5 Passed F0</td><td><u>Uni</u></td></dp>	ar array>) dodules: coseconds): 7.2  Precision DP SP  Precision	+ 11.6n, where array.  How R2 →	n = # of element  Passed Oth element R5 Passed F0	<u>Uni</u>

Algorithm: An accumulator (F0, F1) is initialized to zero. Each element of the array is added to the accumulator in a loop based upon the array size.

	EMAX	<u> </u>
HAL/S-FC LIBE	RARY ROUTINE DESCRIPTION	
Source Member Name: EMAX	Size of Code Area: 8	Hw
Stack Requirement: 0	Hw Data CSECT Size: 0	Hw
X Intrinsic	<pre>Procedure</pre>	
Other Library Modules Reference	ed: None	
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: EMAX		
Function: Finds maximum value	in a single precision scalar array.	
Compiler emitted code for MAX( <sp array="" scalar="">)</sp>	HAD, D CONSCILLOR OF THE POLICY OF THE POLIC	
Other Library Modules:		
Other Library Modules:		
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array)	9.8 + 10.8m + 12.2L, where m = # of; L = # of times CURRMAX changes; and -1.	
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments:	; L = # of times CURRMAX changes; and -1.	E <u>Uni</u>
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments:	; L = # of times CURRMAX changes; and -1.	
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments: Type Precision	; L = # of times CURRMAX changes; and on How Passed	
Execution Time (microseconds): times CURRMAX does not change: M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results:	; L = # of times CURRMAX changes; and $-1$ .  on How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R5	<u>uni</u>
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results: Type Precision	$L = \#$ of times CURRMAX changes; and $-1$ .  On $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$ R5  On $\frac{\text{How Passed}}{\text{R5}}$	<u>Uni</u>
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results:	; L = # of times CURRMAX changes; and $-1$ .  on How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R5	<u>Uni</u>
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments:	the property of times CURRMAX changes; and $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$ and $\frac{\text{How Passed}}{\text{R5}}$	<u>Uni</u>
Execution Time (microseconds): times CURRMAX does not change: M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results: Type Precision Scalar SP	$L = \#$ of times CURRMAX changes; and $-1$ .  On $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$ R5  On $\frac{\text{How Passed}}{\text{R5}}$	
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results: Type Precision Scalar SP  Errors Detected: Error #	$L = \#$ of times CURRMAX changes; and $-1$ .  How Passed  R2 $\rightarrow$ 0 <sup>th</sup> element  R5  On  How Passed  F0  Cause  Fixup	<u>Uni</u>
Execution Time (microseconds): times CURRMAX does not change; M+L = (# of elements in array) Input Arguments: Type Precision Scalar array SP Integer(size) SP  Output Results: Type Precision Scalar SP  Errors Detected: Error #	the property of times CURRMAX changes; and $\frac{\text{How Passed}}{\text{R2} \rightarrow 0^{\text{th}} \text{ element}}$ and $\frac{\text{How Passed}}{\text{R5}}$	<u>Uni</u>

Algorithm: Same as DMAX except that operations are all single precision.

REPRODUCEDUATY OF THE ORIGINAL PAGE 1. BUCK

HAT./S-FC T.TRRARY	•		
imi, b 10 litbionit	ROUTINE DESCRIPTION		
Source Member Name: EMIN	Size of Code Area:	8	Hw
Stack Requirement: 0 Hw	Data CSECT Size: _	0	_ Hw
Intrinsic	<pre>Procedure</pre>		
Other Library Modules Referenced:	None.		· · · · ·
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: EMIN	one was the same of the same		
Function: Finds minimum value in a	n array of single pr	ecision scalar.	
Invoked by:			
Compiler emitted code for HAL,	/S construct of the	form.	
Table 2	's construct or the	LOIM	
MIN( <sp array="" scalar="">)</sp>			
Other Library Modules:			
Execution Time (microseconds): 9.	8 + 10.8m + 12.2L, v	here m = # of t	imes
CURRMIN does not change; L = # of	times CURRMIN chance	es: and M+L = 0	(# of
elements in array)-1.			
Input Arguments:			
Type Precision	How Pa		Un
	$R2 \rightarrow 0$ th	element	
Scalar array SP			
	R5		
Scalar array SP	R5		
Scalar array SP Integer(size) SP	R5		
Scalar array SP Integer(size) SP Output Results:			IJņ
Scalar array SP Integer(size) SP  Output Results: Type Precision	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP Output Results:		ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected:	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected:	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected:	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected:	How Pa	ssed	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected: Error #	How Pa	<u>Fixup</u>	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results:  Type Precision Scalar SP  Errors Detected:	How Pa	<u>Fixup</u>	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results: Type Precision Scalar SP  Errors Detected: Error #	How Pa	<u>Fixup</u>	<u>Un</u>
Scalar array SP Integer(size) SP  Output Results: Type Precision Scalar SP  Errors Detected: Error #	How Pa FO Cause	Fixup  FO, F1.	

		EPROD	
'C LIBRARY	ROUTINE DESCRIPTION		
EPROD	Size of Code Area:	10	F
0 Hw	Data CSECT Size: _	0	I
	Procedure		
erenced: _	None.		
EDDOD			
EFROD	-		
uct of ele	ments of a single p	recision scalar	ar
In for HAT	's construct of the	forme	
	5 Construct of the	LOTH:	
ray>)			
es:			
12	2n + 1 6 if needings		L -
mas): 13.	2n + 4.6 if produce	s is not zero, w	ne
e array. 1	3.2m + 1.4 if produc	t is zero, wher	e :
∍presentati	on of the array of	the first zero	el
		_	
cecision	How Pa	ssed	
SD	p2 → nth	element	
סר		R5	
recision			
	11 Ti-		
FECTSTOIL	How Pa	<u>issea</u>	
SP	How Pa		
<del></del>			
<del></del>	F	<b>3</b>	
<del></del>			
<del></del>	F	<b>3</b>	
<del></del>	F	<b>3</b>	
SP	<u>Cause</u>	Fixup	
SP	F	Fixup	
SP	<u>Cause</u>	Fixup	
	EPROD  O Hw  Gerenced:  EPROD  uct of elected for HAL/ray>)  es:  onds): 13.  e array. 1  epresentation	EPROD Size of Code Area:  O Hw Data CSECT Size: Procedure  EPROD  uct of elements of a single product of the ray>)  es:  onds): 13.2n + 4.6 if produce array. 13.2m + 1.4 if produce expresentation of the array of the recision  SP R2 + 0th SP 1	O Hw Data CSECT Size: 0  Procedure  EPROD  uct of elements of a single precision scalar  de for HAL/S construct of the form: ray>)  es:  onds): 13.2n + 4.6 if produce is not zero, where array. 13.2m + 1.4 if product is zero, where apresentation of the array of the first zero recision  How Passed

			E	SUM	
HAL/S	-FC LIBRARY	ROUTINE DESCRI	PTION		
 Source Member Name:	ESUM	Size of Code	Area: 6		Hw
Stack Requirement:	O Hw	Data CSECT Si	ze:	0	Hw
▼ Intrinsic		☐ Procedur	е		
Other Library Modules R	eferenced:	None			
ENTRY POINT DESCRIPTION	S				
Primary Entry Name:	ESUM				
Function: Calculates su	m of elemen	ts in a single p	recision so	alar array	•
Invoked by:    Compiler emitted c     SUM( <sp library="" modu<="" other="" scalar="" td=""  =""><td>array&gt;) les:</td><td></td><td></td><td>lements in</td><td>the arra</td></sp>	array>) les:			lements in	the arra
Input Arguments:					
	Precision SP SP		ow Passed → O <sup>th</sup> eleme R5	ent	Units - -
Output Results:					
<u>Type</u> Scalar	Precision SP	<u>H</u>	Passed FO		<u>Units</u>
Errors Detected:	S.		FO		
Error #		Cause		Fixup	
		Call: R2, R4,	R5, F0, F1		
Algorithm: Same as I	DOUM.		法国国际 医动脉管		

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

		XAMH	
HAL/S-FC LIBR	ARY ROUTINE DESCRIPTIO	N	
Source Member Name: HMAX	Size of Code Area		_ Hw
	Hw Data CSECT Size:	0	_ Hw
Intrinsic	Procedure		
<b>Th</b> -	a. None.		
Other Library Modules Reference	a: Notice		
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: HMAX			
Function: Finds maximum valu	e in a single precisio	on integer array.	
Invoked by:  X Compiler emitted code for  MAX( <sp array="" integer="">)  Other Library Modules:  Execution Time (microseconds):</sp>	11 0 1 7 9m + 9 2k.	where m = # of t	imes
CURRMAX does not change, A	# of times CURRMAX ch	anges; and m+k =	(# OT
elements in array)-1. Input Arguments:	How	Passed	Uni
Type Precisi	R2 → 0	th element	· · · · · · · · · · · · · · · · · · ·
Integer array SP Integer(size) SP		R5	<del>-</del>
Output Results:	ion <u>How</u>	Passed	Uni
Type Freeze SP		R5	
Errors Detected: Error #	<u>Cause</u>	<u>Fixup</u>	
Comments: Registers Unsafe A	Across Call: R2, R4, R	b, Ro.	

Algorithm: Same as DMAX, except that all operations deal with halfword integers.

	HMIN	
	4 A & A & A & A & A & A & A & A & A & A	
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION	
	Source Member Name: HMIN Size of Code Area: 8	- Hw
	Stack Requirement: 0 Hw Data CSECT Size: 0.	Hw
	X Intrinsic Procedure	
	Other Library Modules Referenced: None	
	ENTRY POINT DESCRIPTIONS	
	Primary Entry Name: HMIN	
	Function: Finds minimum value in a single precision integer array,	
		\$ 14 m
	Invoked by:  X Compiler emitted code for HAL/S construct of the form:  MIN( <sp array="" integer="">)</sp>	
	Other Library Modules:	
	Execution Time (microseconds): 11.0 + 7.8m + 9.2k, where m = # of to CURRMIN does not change; k = # of times CURRMIN changes; and m+k = elements in array)-1.	imes (# of
	Input Arguments:  Type Precision How Passed	Unit
	Integer array SP R2 $\rightarrow$ Oth element Integer(size) SP R5	<del></del>
	Output Results:	
	Type Precision How Passed Integer SP R5	<u>Unit</u>
	Errors Detected: Error # Cause Fixup	
	Comments: Registers Unsafe Across Call: R2, R4, R5, R6.	
. :: 1	Algorithm: Same as DMIN, except that operations are for halfword in	

			HPROD	
на	L/S-FC LIBRARY	ROUTINE DESCRIPT	CON	
Source Member Name:	HPROD	Size of Gode Are	ea: 12	HW
Stack Requirement:	0 <sub>Hw</sub>	Data CSECT Size:	. 0	Hw
X Intrins	lc	☐ Procedure		
Other Library Modules	Referenced:			
ENTRY POINT DESCRIPT	LONS	,		
Primary Entry Name:	HPROD			
Function: Calculates	product of el	ements in a single	precision inte	ger array.
Invoked by:				
X Compiler emitted	d code for HAI	/S construct of the	ne form:	
PROD( <sf inte<="" td=""><td>ger array&gt;)</td><td></td><td></td><td></td></sf>	ger array>)			
Other Library Mo	dules:			
Execution Time (micro	seconds): 1	2.4n + 5.8 if prod	uct is not zero.	
				where
<pre>n = # of elements i into the linear rep Input Arguments:</pre>	n array. 12.4 resentation of	4m + 2.2 if produce the array of the	t is zero, where	obni = m
<pre>n = # of elements i into the linear rep Input Arguments:</pre>	n array. 12.4 resentation of Precision	t the array of the	t is zero, where	e m = inde ment.
Input Arguments:	resentation o	t the array of the	t is zero, where first zero elem Passed	m = inde ment.
Into the linear rep Input Arguments:  Type	Precision of	t the array of the	t is zero, where first zero elem	m = indo
Into the linear rep Input Arguments:  Type  Integer array	Precision of SP	t the array of the	t is zero, where first zero elem <u>Passed</u> O <sup>th</sup> element	e m = inde ment.
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type	Precision SP SP Precision	t the array of the $\frac{\text{How}}{\text{R2}}$	t is zero, where first zero elem  Passed  Oth element  R5  Passed	m = inde ment.
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:	Precision of SP SP	t the array of the $\frac{\text{How}}{\text{R2}}$	t is zero, where first zero elem  Passed  Oth element  R5	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type	Precision SP SP Precision	t the array of the $\frac{\text{How}}{\text{R2}}$	t is zero, where first zero elem  Passed  Oth element  R5  Passed	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type Integer Integer	Precision SP SP Precision	t the array of the $\frac{\text{How}}{\text{R2}}$	t is zero, where first zero elem  Passed  Oth element  R5  Passed	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type Integer  Integer  Errors Detected:	Precision SP SP Precision	the array of the How R2 $\rightarrow$ How	t is zero, where first zero elem  Passed  Oth element  R5  Passed  R5	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type Integer  Integer  Errors Detected:	Precision SP SP Precision	the array of the How R2 $\rightarrow$ How	t is zero, where first zero elem  Passed  Oth element  R5  Passed  R5	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type Integer  Errors Detected:  Error #	Precision SP SP Precision	the array of the  How R2 →  How	t is zero, where first zero element R5  Passed  Passed  Passed  R5  Passed  R5	e m = inde ment. Unit
Into the linear rep Input Arguments:  Type  Integer array Integer(size)  Output Results:  Type Integer  Errors Detected:  Error #	Precision SP SP Precision SP SP	the array of the  How R2 →  How  Cause	t is zero, where first zero element R5  Passed  Passed  Passed  R5  Passed  R5	e m = indenent. Unit

				HSUM	•
			DOMESTIC DECORTOR		
	HAJ	L/S-FC LIBRARY	ROUTINE DESCRIPTION		<u>.</u> .
٤	Source Member Name:	HSUM	Size of Code Area:	6	Hw
٤	Stack Requirement:	0 <u>H</u> w	Data CSECT Size: _	0	Hw
	X Intrins	ic	☐ Procedure		
(	Other Library Module	s Referenced:	None.		, <u></u>
<u> </u>	ENTRY POINT DESCRIPT	IONS			
	Primary Entry Name:		<del>, , , , , , , , , , , , , , , , , , , </del>		
	Function: Calculate	s sum of eleme	nts in <b>a</b> single prec	ision integer	array.
	Invoked by:  Compiler emitte	d code for HAI	L/S construct of the	form:	
	SUM(≺SP inte				
	Other Library M	lodules:			
	Orner minrary				
	Execution Time (micr	roseconds): 4.	4 + 5.4n, where $n = 3$	of elements:	in the a
	DACCHETOH TIME (MIC)				
	Input Arguments:		How P	hesse	Un:
	<u>Type</u>	Precision		th element	
	Integer array Integer(size)	SP SP	R		
	Output Results:	Danama and an	How P	assed	Un
	<u>Type</u> Integer	Precision SP	R!		
	Errors Detected:		Cauca	Fixup	
	Error #		Cause		

Algorithm: Same as DSUM.

			XAMT	
HAL	/S-FC LIBRARY	ROUTINE DES	SCRIPTION	
Source Member Name: _	IMAX	Size of Co	ode Area: 8	Hw
Stack Requirement:	0 Hw	Data CSEC	r Size: 0	Hw
X Intrinsi		☐ Proc	edure	
Other Library Modules		None.		
ENTRY POINT DESCRIPTI	IONS			
Primary Entry Name:	IMAX			
Function: Finds max	kimum value ir	a double p	recision integer	ar <b>ra</b> y.
Invoked by:  [x] Compiler emitted	d code for HA	L/S constru	et of the form:	
MAX( <dp intege<="" td=""><td>er array&gt;)</td><td></td><td></td><td></td></dp>	er array>)			
Other Library M	odules:			
Execution Time (micr CURRMAX does not c elements in array-	nange, x - , m '	1.1 + 7.8m + of times CUF	4.3k, where m = RRMAX changes, and	# of times 1 m+k = # of
Input Arguments: Type	Precision		How Passed	<u>Unit</u>
Integer array Integer(size)	DP SP		$R2 \rightarrow 0^{th}$ elemen R5	
Output Results: Type	Precision		How Passed	Unit
Integer	DP		<b>R</b> 5	
Errors Detected: Error #		Cause		Fixup
Comments: Register	s Unsafe Acro	ss Call: R2	, R4, R5, R6.	

Algorithm: Same as DMAX, except that all operations are on fullword integers.

REPRODUCIBILITY OF THE ORIGINAL PAGE IN SOUTH

		IMIN	
	HAL/S-FC LIBRARY ROUT	INE DESCRIPTION	
Source Member Na	me: IMIN Siz	e of Code Area: 8	1
Stack Requiremen	it: 0 Hw Dat	a CSECT Size: 0	]
<b>⊠</b> Int	rinsic [	] Procedure	
Other Library Mo	dules Referenced:	None.	
ENTRY POINT DESC			
ENTRI POINI DESC	.RIF 110N3		
Primary Entry Na	me: IMIN		
Function: Find	ds minimum value in a do	uble precision integer arra	У
Invoked by:			
Compiler em	nitted code for HAL/S co	enstruct of the form:	
MTN(SDP i	integer array/)		
MIN( <dp 1<="" td=""><td>integer array&gt;)</td><td></td><td></td></dp>	integer array>)		
	ary Modules:		
Other Libra	ary Modules:		
Other Libra	ary Modules: (microseconds): 11.1 +	7.8m + 9.3k, where m = # of	times
Other Libra  Execution Time ( does not change array -1.	microseconds): 11.1 + e, k = # of times CURRMI	7.8m + 9.3k, where $m = \#$ of N changes, and $m+k = \#$ of e	times
Other Libra  Execution Time ( does not change array -1.  Input Arguments:	ery Modules: (microseconds): 11.1 + e, k = # of times CURRMI	N changes, and $m+k = \#$ of e	times lemen
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type	microseconds): 11.1 + e, k = # of times CURRMI  Precision	7.8m + 9.3k, where m = # of N changes, and m+k = # of e  How Passed R2 + Oth element	times lemen
Other Libra  Execution Time ( does not change array -1.  Input Arguments:	microseconds): 11.1 + e, k = # of times CURRMI  r y  Precision DF	N changes, and m+k = # of e  How Passed	times
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer(size)	microseconds): 11.1 + e, k = # of times CURRMI  r y  Precision DF	N changes, and m+k = # of e  How Passed  R2 + Oth element	times
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array	microseconds): 11.1 + e, k = # of times CURRMI  r y  Precision DF	N changes, and m+k = # of e  How Passed  R2 + Oth element	times
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer(size)  Output Results:	microseconds): 11.1 + e, k = # of times CURRMI  Precision DP SP	N changes, and m+k = # of e  How Passed  R2 + Oth element  R5	times
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer(size)  Output Results:  Type	microseconds): 11.1 + e, k = # of times CURRMI  Precision PP SP  Precision DP DP	How Passed R2 + Oth element R5  How Passed R5	emen!
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer(size)  Output Results:  Type Integer	microseconds): 11.1 + e, k = # of times CURRMI  precision pp SP  Precision DP DP SP	How Passed  R2 - Oth element  R5  How Passed	elemen
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer (size)  Output Results:  Type Integer  Integer  Integer  Integer	microseconds): 11.1 + e, k = # of times CURRMI  precision pp SP  Precision DP DP SP	How Passed R2 + Oth element R5  How Passed R5	emen
Other Libra  Execution Time ( does not change array -1.  Input Arguments:  Type Integer array Integer(size)  Output Results:  Type Integer  Errors Detected: Error #	microseconds): 11.1 + e, k = # of times CURRMI  precision pp SP  Precision DP DP SP	How Passed R2 + Oth element R5  How Passed R5  How Passed Fixu	elemen

5-282

			PROD	
	HAL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name	: IPROD Siz	e of Code Area: _	22	Hw
Stack Requirement:	0 Hw Dat	a CSECT Size:	0	Hw
🔀 Intri	nsic [	Procedure		
Other Library Modul	les Referenced: N	lone,		
ENTRY POINT DESCRIE	PTIONS			
Primary Entry Name:	IPROD			
Function: Calculates	product of elements	in a double preci	sion intege	r array.
Invoked by:	ced code for HAL/S co eger array>)	nstruct of the fo	rm;	
Invoked by:  Compiler emitt  PROD ( <dp (mice<="" execution="" int="" library="" other="" th="" time=""><th>eger array&gt;) Modules: roseconds): 17.0L +</th><th>21.6m + 5.8 if pr</th><th>oduct is no</th><th></th></dp>	eger array>) Modules: roseconds): 17.0L +	21.6m + 5.8 if pr	oduct is no	
Invoked by:    X Compiler emitt   PROD( <dp (mic="" ab<="" and="" are="" as="" elemen="" execution="" int="" l="" l+m="#" library="" m="" of="" other="" positive="" th="" time=""  =""><th>eger array&gt;) Modules:</th><th>21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if</th><th>oduct is no ive interme product is</th><th>diate pro</th></dp>	eger array>) Modules:	21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if	oduct is no ive interme product is	diate pro
Invoked by:  X Compiler emitt  PROD( <dp (mic="" elemen<="" execution="" int="" l="#" l+m="#" library="" of="" other="" positive="" td="" time=""><td>eger array&gt;)  Modules:  roseconds): 17.0L +  intermediate product  ts in array. 17.0L +</td><td>21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if to linear represer tent)-1.</td><td>coduct is no live interme product is litation of t</td><td>diate pro</td></dp>	eger array>)  Modules:  roseconds): 17.0L +  intermediate product  ts in array. 17.0L +	21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if to linear represer tent)-1.	coduct is no live interme product is litation of t	diate pro
Invoked by:    X   Compiler emitty	<pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int</pre>	21.6m + 5.8 if property of property of property of the second prope	coduct is no live interme product is litation of t	diate pro not zero, he array
Invoked by:    X Compiler emitt   PROD( <dp (mid="" ab="" and="" are="" arguments:="" as="" elemen="" execution="" input="" int="" l="" l+m="#" library="" m="" of="" other="" positive="" td="" time="" type<=""  =""><td><pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int     the first zero elem</pre></td><td>21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if to linear represer tent)-1.</td><td>coduct is no live interme product is litation of t</td><td>diate pro not zero, he array</td></dp>	<pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int     the first zero elem</pre>	21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if to linear represer tent)-1.	coduct is no live interme product is litation of t	diate pro not zero, he array
Invoked by:    X   Compiler emitty     PROD( <dp (mic="" ab="" and="" are="" arguments:="" array<="" as="" elemen="" execution="" input="" integer="" into="" l="" l+m="#" library="" m="" of="" other="" positive="" td="" time="" type=""  =""><td><pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int</pre></td><td>21.6m + 5.8 if property of property of property of the second prope</td><td>coduct is no live interme product is litation of t</td><td>diate pro not zero, he array</td></dp>	<pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int</pre>	21.6m + 5.8 if property of property of property of the second prope	coduct is no live interme product is litation of t	diate pro not zero, he array
Invoked by:    X Compiler emitt	<pre>eger array&gt;) Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int</pre>	21.6m + 5.8 if property of property of property of the second prope	coduct is no zive interme product is station of t ed ement	diate pro not zero, he array Unit
Invoked by:    X Compiler emitt   PROD( <dp (mic="" (size)="" ab="" and="" are="" arguments:="" array="" as="" elemen="" execution="" input="" int="" integer="" l="" l+m="#" library="" m="" of="" other="" output="" positive="" results:<="" td="" time="" type=""  =""><td><pre>Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int       the first zero elem</pre></td><td>21.6m + 5.8 if properties; m = # of negative 21.6m + 19.6 if to linear representation of the second content.  How Pass  R2 <math>\rightarrow</math> 0th e1  R5</td><td>coduct is no zive interme product is station of t ed ement</td><td>diate pro not zero, he array Unit</td></dp>	<pre>Modules: roseconds): 17.0L +   intermediate product ts in array. 17.0L +   ove, L+m = (index int       the first zero elem</pre>	21.6m + 5.8 if properties; m = # of negative 21.6m + 19.6 if to linear representation of the second content.  How Pass  R2 $\rightarrow$ 0th e1  R5	coduct is no zive interme product is station of t ed ement	diate pro not zero, he array Unit
Invoked by:    X   Compiler emitty	Modules: roseconds): 17.0L + intermediate product ts in array. 17.0L + ove, L+m = (index int the first zero elem  Precision DP SP  Precision	21.6m + 5.8 if pr s; m = # of negat 21.6m + 19.6 if to linear representent)-1. How Pass R2 + 0th el R5	coduct is no zive interme product is station of t ed ement	diate pro not zero, he array

Comments: Registers Unsafe Across Call: R2, R4, R5, R6, R7.

Algorithm: Same basic algorithm as DPROD, however, special detection of an overflow condition is performed: For fullword integer multiplication, the overflow indicator is only set when -l is multiplied by -l. The result after each multiplication is checked for an overflow by testing the first 32 bits of the 64 bit result for all zeros or ones. If the result does overflow 32 bits, then a fixed point overflow is forced by adding a very large number to the first register of the pair (the register with the overflowing bits).

			ISUM	
	AL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	ISUM Siz	e of Code Area:	6	Hw
				<del></del>
tack Requirement:	O Hw Dat	a CSECT Size:		_ Hw
X Intrins	sic [	Procedure	.1	
Other Library Module	es Referenced: None	9.		
ENTRY POINT DESCRIPT	rions			<del>r dadişə</del> ;
Primary Entry Name:	ISUM			
			and the second second	
Function: Calculates	s sum of elements in	a double precision	integer arra	ay.
Invoked by:		•		
ly Compiler emitte	ed code for HAT/S co	nstruct of the for	m:	
	eger array>)			
	eger array>)			
SUM( <dp int<="" td=""><td>eger array&gt;)</td><td></td><td></td><td></td></dp>	eger array>)			
SUM( <dp int<="" th=""><th>eger array&gt;) Modules:</th><th></th><th></th><th></th></dp>	eger array>) Modules:			
SUM( <dp int<="" th=""><th>eger array&gt;) Modules:</th><th></th><th></th><th>n arra</th></dp>	eger array>) Modules:			n arra
SUM( <dp (mice<="" execution="" int="" library="" m="" other="" th="" time=""><th>eger array&gt;) Modules:</th><th></th><th></th><th>n arra</th></dp>	eger array>) Modules:			n arra
SUM( <dp (mich="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>eger array&gt;) Modules: Mosecónds): 4.4 + 5</td><td>5.4n, where <math>n = # o</math></td><td>of elements in</td><td></td></dp>	eger array>) Modules: Mosecónds): 4.4 + 5	5.4n, where $n = # o$	of elements in	
SUM( <dp (mich="" arguments:="" execution="" input="" int="" library="" m="" other="" td="" time="" type<=""><td>eger array&gt;) Modules:</td><td>5.4n, where n = # o</td><td>of elements in</td><td></td></dp>	eger array>) Modules:	5.4n, where n = # o	of elements in	
SUM( <dp (mich="" arguments:="" array<="" execution="" input="" int="" integer="" library="" m="" other="" td="" time="" type=""><td>eger array&gt;) Modules: roseconds): 4.4 + 5  Precision DP</td><td>5.4n, where <math>n = # o</math></td><td>of elements in</td><td></td></dp>	eger array>) Modules: roseconds): 4.4 + 5  Precision DP	5.4n, where $n = # o$	of elements in	
SUM( <dp (mich="" arguments:="" execution="" input="" int="" library="" m="" other="" td="" time="" type<=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision</td><td>5.4n, where n = # o</td><td>of elements in</td><td>n arra Uni</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision	5.4n, where n = # o	of elements in	n arra Uni
SUM( <dp (mich="" arguments:="" array="" execution="" input="" int="" integer="" integer(size)<="" library="" m="" other="" td="" time="" type=""><td>eger array&gt;) Modules: roseconds): 4.4 + 5  Precision DP</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow</math> 0<sup>th</sup> el</td><td>of elements in</td><td></td></dp>	eger array>) Modules: roseconds): 4.4 + 5  Precision DP	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow$ 0 <sup>th</sup> el	of elements in	
SUM( <dp (mice="" arguments:="" array="" execution="" input="" int="" integer="" integer(size)="" library="" m="" other="" output="" results:<="" td="" time="" type=""><td>eger array&gt;) Modules: roseconds): 4.4 + 5  Precision DP</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow</math> 0<sup>th</sup> el</td><td>of elements in description of elements in description of the descripti</td><td>Uni - -</td></dp>	eger array>) Modules: roseconds): 4.4 + 5  Precision DP	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow$ 0 <sup>th</sup> el	of elements in description of elements in description of the descripti	Uni - -
SUM( <dp (micro="" arguments:="" array="" execution="" input="" int="" integer="" integer(size)<="" library="" m="" other="" td="" time="" type=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP SP</td><td>5.4n, where <math>n = \# c</math> <math display="block">\frac{\text{How Passe}}{\text{R2} \to 0^{\text{th}} \text{ el}}</math></td><td>of elements in description of elements in description of the descripti</td><td>Uni -</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP SP	5.4n, where $n = \# c$ $\frac{\text{How Passe}}{\text{R2} \to 0^{\text{th}} \text{ el}}$	of elements in description of elements in description of the descripti	Uni -
SUM( <dp (mice="" arguments:="" array="" execution="" input="" int="" integer="" integer(size)="" integer<="" library="" m="" other="" output="" results:="" td="" time="" type=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision</td><td>5.4n, where <math>n = \# c</math>  How Passes  R2 <math>\rightarrow 0^{th}</math> el  R5</td><td>of elements in description of elements in description of the descripti</td><td>Uni</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision	5.4n, where $n = \# c$ How Passes  R2 $\rightarrow 0^{th}$ el  R5	of elements in description of elements in description of the descripti	Uni
SUM( <dp (micro="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP</td><td>5.4n, where <math>n = \# c</math>  How Passer  R2 <math>\rightarrow 0^{\text{th}}</math> el.  R5  How Passer  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni - -</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP	5.4n, where $n = \# c$ How Passer  R2 $\rightarrow 0^{\text{th}}$ el.  R5  How Passer  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni - -
SUM( <dp (mice="" arguments:="" array="" execution="" input="" int="" integer="" integer(size)="" integer<="" library="" m="" other="" output="" results:="" td="" time="" type=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision</td><td>5.4n, where <math>n = \# c</math>  How Passer  R2 <math>\rightarrow 0^{\text{th}}</math> el.  R5  How Passer  R5</td><td>of elements in description of elements in description of the descripti</td><td></td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision	5.4n, where $n = \# c$ How Passer  R2 $\rightarrow 0^{\text{th}}$ el.  R5  How Passer  R5	of elements in description of elements in description of the descripti	
SUM( <dp (micro="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP</td><td>5.4n, where <math>n = \# c</math>  How Passer  R2 <math>\rightarrow 0^{\text{th}}</math> el.  R5  How Passer  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni -</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP	5.4n, where $n = \# c$ How Passer  R2 $\rightarrow 0^{\text{th}}$ el.  R5  How Passer  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni -
SUM( <dp #<="" (micro="" arguments:="" array="" detected:="" error="" errors="" execution="" input="" int="" integer="" integer(size)="" library="" m="" other="" output="" results:="" td="" time="" type=""><td>Precision DP Precision DP SP  Precision DP Cau</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow 0^{th}</math> el  R5  How Passes  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni - -</td></dp>	Precision DP Precision DP SP  Precision DP Cau	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow 0^{th}$ el  R5  How Passes  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni - -
SUM( <dp (micro="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>eger array&gt;)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow 0^{th}</math> el  R5  How Passes  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni - -</td></dp>	eger array>)  Modules:  roseconds): 4.4 + 5  Precision  DP  SP  Precision  DP	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow 0^{th}$ el  R5  How Passes  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni - -
SUM( <dp (micro="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>Precision DP Precision DP SP  Precision DP Cau</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow 0^{th}</math> el  R5  How Passes  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni.</td></dp>	Precision DP Precision DP SP  Precision DP Cau	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow 0^{th}$ el  R5  How Passes  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni.
SUM( <dp (micro="" arguments:<="" execution="" input="" int="" library="" m="" other="" td="" time=""><td>Precision DP Precision DP SP  Precision DP Cau</td><td>5.4n, where <math>n = \# c</math>  How Passe  R2 <math>\rightarrow 0^{th}</math> el  R5  How Passes  R5</td><td>of elements in description of element description of elements in the elements described and element described and elements described and</td><td>Uni</td></dp>	Precision DP Precision DP SP  Precision DP Cau	5.4n, where $n = \# c$ How Passe  R2 $\rightarrow 0^{th}$ el  R5  How Passes  R5	of elements in description of element description of elements in the elements described and element described and elements described and	Uni

EFFWHEICEHETY OF THE URLEWIST, PARTY IN THE

## 5.3.6 Miscellaneous Routine Descriptions

This subsection presents those routines which do not fall easily into the previous five sections. These encompass conversion routines as well as "service" routines used by other library members.

	Ĺ	Dr.
	٠	
′ :		10
703	ŀ	43

			BTOC	_
HAL/	S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	BTOC	Size of Code Area:	28	Hw Hw
Stack Requirement:		Data CSECT Size:	0	Hw
X Intrinsic	1	☐ Procedure		
Other Library Modules	Referenced:	None.		
ENTRY POINT DESCRIPTION Primary Entry Name:				
Function: Conversion		ta to character data.		
Invoked by:  Compiler emitted  CHARACTER  BIN  Other Library Mo	Lt string>).	/S construct of the	form;	
Execution Time (micro	seconds):	161.0 (for 16-bit	string)	
Input Arguments: <u>Type</u> Bit  Integer(length)	Precision SP	How Po R5 R6	assed	Units - -
Output Results: <u>Type</u> Character	Precision -		assed criptor	Units -
Errors Detected: Error #		Cause	Fixup	

Comments: Registers Unsafe Across Call: Rl,R2,R3,R4,R5,R6,R7.

Algorithm: First, unwanted bits are shifted out of the string, using the length argument. Then, bits are shifted one by one out of the top of the R5 into the bottom of R4, where they are shifted to bit positions 15 and 31 and converted to character format. The output string is stored halfword by halfword, with the length taken directly from the input length.

			CSHAPQ	
HAL	/s-FC LIBRARY F	COUTINE DESCRIPTION		
Source Member Name:	CSHAPQ	Size of Code Area:	40	Hw
Stack Requirement:	18 Hw	Data CSECT Size:	4	Hw
☐ Intrinsi	C	▼ Procedure		· T
Other Library Modules	Referenced: _	стон сто	I,CTOE,CTOD	
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name: _	CSHAPQ			
	ayed character ype and precisi	data to arrayed nume on.	ric data of	
Invoked by:  \[ \text{X} \] Compiler emitted  INTEGER (CA),	code for HAL/	S construct of the f $_{ m N}^{ m (CA)}$ , SCALAR $_{ m N}^{ m (CA)}$ ,	orm; SCALAR	N(CA),
Other Library Mo	where CA is	a character array of	length n of ACTER(m).	
Execution Time (micro	seconds): (Se	e next page).		
Input Arguments: Type	Precision	How Pas	seđ	Units
character array(n		$R4 \rightarrow 1st e$	Lement	· · · · · · · · · · · · · · · · · · ·
<pre>integer(n) integer(type *) integer(m) Output Results:</pre>	SP	R5 R6 R7		
Type	Precision	How Pas	sed	Units
array(n)	type	R2		
Errors Detected:				
Error #		Cause	<u>Fixup</u>	
Comments: Regista	type*: 0 → 1 →	ss Call: F0,F1,F2,F H conversion I conversion E conversion	3,F4,F5	
Algorithm:		D conversion	보기들이 다른하는 그리고 원칙적 교육의 기계 전기 기계 기계를	
the type code as	an index and c	l loops is loaded fr ontrol is passed to in action: a call i	that loop. Th	

The address of one of 4 internal loops is loaded from a table using the type code as an index and control is passed to that loop. The four internal loops are similar in action: a call is made to the appropriate character conversion routine (CTOH, CTOI, CTOE, CTOD) followed by the appropriate store (STH, ST, STE, STED) into the result array, followed by instructions to bump both the character and result array pointers, looping on n.

# REPRODUCIBILITY OF THE ORIGINAL PAGE TO EXTRE

CSHAPQ

CSHAPQ

Execution Time (microseconds):

For halfword INTEGER conversion:

25.2 + 
$$\sum_{k=1}^{n}$$
 (19.6 + CTOH<sub>K</sub>)

where  $CTOH_{K}$  = time in CTOH for the  $K^{th}$  conversion.

For fullword INTEGER conversion:

24.8 + 
$$\sum_{k=1}^{n}$$
 (20.2 + CTOI<sub>K</sub>)

where  $\mathtt{CTOI}_{K}$  = time in CTOI for the  $\mathtt{K}^{\mathtt{th}}$  conversion.

For fullword SCALAR conversion:

25.2 + 
$$\Sigma$$
 (19.6 + CTOE<sub>K</sub>)  
k=1

where  $\mathtt{CTOE}_{K}$  = time in CTOE for the  $\mathtt{K}^{\mathtt{th}}$  conversion.

For double-word SCALAR conversion:

22.8 + 
$$\sum_{k=1}^{n}$$
 (22.8 + CTOD<sub>K</sub>)

where  $\mathtt{CTOD}_{K}$  = time in CTOD for the  $\mathtt{K}^{\mathsf{th}}$  conversion.

			CS	ID	
•	HAL/S-FC LIBRAR	Y ROUTINE DES	SCRIPTION		
Source Member N	lame: CSLD	Size of Co	ode Area:	246	_ Hw
Stack Requireme	ent: 22 Hw	Data CSEC	r Size:	4	Hw
☐ Ir	ntrinsic	X Pro	cedure		
Other Library M	Modules Referenced:	None.			
ENTRY POINT DES	CRIPTIONS				
Primary Entry N	Name: CSLD				
Function: Loads	bit pattern of a	character st	ring.		
B=SI	Precision	a bit strin	g and C is a	character s ) > 4)	tring. Units
Output Results Type			How Passe		Units
bit string		•	R5		
Errors Detecte Error #		Cause		Fixup	
Comments:	If input string is	null, the 0	bit string is	s returned;	no error.
Register	s Unsafe Across Cal	.1: R5,F0,F1.			
charact the res	The first character er width is set to t, see the descript er partition check	the current ion under em	length of the	e string. Fo fter the	<b>r</b>

CSLD

Secondary Entry Name: CSLDP	
Function: Loads bit pattern of a partitioned character string.	
Invoked by:	
<pre>Compiler emitted code for HAL/S construct of the form: BIT_STRING = SUBBIT(CHAR_STRING_WIDTH AT FIRST)</pre>	
Other library modules:	
Execution Time (microseconds): 69.7 + 0.8 (if WIDTH > 4) + 4.0 (if FIRST is even)	
Input Arguments:  Type Precision How Passed  character string R2 + descriptor  integer (first character) SP R5  integer (last character) SP R6	Units
Output Results:  Type Precision How Passed  bit string length(32) R5	<u>Units</u>
Comments: 0 bit string returned if last character 3) If last character specified < first character specified char, ret (with ERROR 17). Registers Unsafe Across Call: R5,F0,F1.	th of string, ngth (string) har < first turn 0 string
Algorithm: The character partition is checked for validty before any else is done. All error 17's are sent during this phase. A The partition width is checked, and if it is < 0, the zero string is returned in R5. If greater than 4, it is set to 4. The address of the halfword containing the first character of the partition is for adding 1/2(1+first character) to the address of the first halfword of the string. This halfword and the next two halfwords are loaded into the low half of R4, and the high and low halves of R5, respectint to the low half of R4, and the high and low halves of the right (sl	ind by itively,

Unwanted bits are masked off the left and shifted off the right (shift

count = 48-8\*width), and the desired bit string is left in R5.

CSLD		

Secondary Entry Name:	CPSLD			
Function: Loads spe	ecified bits of a	character string.		
Invoked by:				
	code for HAL/S co	onstruct of the fo	en:	
Other library mod	lules:	tiga er gjerke gjelfræk Gjerke Gjelfræke er etter er		
Execution Time (micros	seconds): 71.8			
Input Arguments:				
Type character string	Precision	R2 + descr		Units
<pre>integer(first bit) integer(last bit)</pre>	SP SP	R5 R6		
Output Results:				
Type bit string	Precision length(32)	How Pas R5	sed	<u>Units</u>
Errors Detected:				
· Error #	<b></b>	use	Fixup	
30 Su	bbit partition ou		If 1st bit <1, (keep constant) width by adjust If 1st or last last bit of cha set equal to la char string.	partition ing last bit) bit > racter string,
Comments: If input s				
	t string is retur			
. Registers	Unsafe Across Cal	T: K2'E0'ET"		

Registers Unsafe Across Call: R5,F0,F1.

Algorithm: The subbit partition is tested for validity before anything else is done. All ERROR 30's are sent during these tests. 4 halfwords containing the required partition are loaded into a register pair. Unwanted bits are shifted off the top (left shift count = first bit-1), and the bottom (right shift count = 64-width), leaving the required string in R5.

CSLD

Secondary Entry Name: CPSLDP

Function: Loads selected bits of a partitioned character string.

#### Invoked by:

Compiler emitted code for HAL/S construct of the form:

BIT\_STRING = SUBBIT\_A TO B (CHAR\_STRING\_C TO D)

Other library modules:

Execution Time (microseconds): 98.6 + 9.2 (if C is even)

Input Arguments:

Type	Precision	How Passed	Units
character string	ing the state of t	R2 → descriptor	
integer(C) (Continued on	SP bottom of this page)	R5	
Output Results:		and the second of the second of the second	
Type	Precision (22)	How Passed	Units
hit string	length (32)	R5	•

#### Errors Detected:

Error #	<u>Cause</u>	Fixup
17	character subscript out of legal range.	(See CSLDP)
30	subbit partition out of legal range.	(See CPSLD)

#### Comments:

Registers Unsafe Across Call: R5,F0,F1.

Algorithm: The character partition is checked for validity, and the 0 bit string is returned if last character < first character. Then the subbit partition is checked, resetting the bit pointer to point 8 bits farther on if the character partition begins in the second character of a halfword. 4 halfwords containing the required partition are loaded into register pair R4-R5. Unwanted bits are shifted off the top (shift count = relative lst bit-1) and the bottom (shift count = 64-bit width), leaving the desired string in R5.

Input Arguments (Con't):

Type	Precision	How Passed	Units
integer(D)	SP	R6	pi dingana di 🖛 i
integer(A   B)	(SP   SP)	R7	

CSLD		
سدر	 	

Secondary Entry Name:	CSST			
Function: Stores a bit	string into a	character st	ring.	
Invoked by:				
Compiler emitted SUBBIT(C) = 1	code for HAL/S B where B is a	construct of BIT string a	f the form: nd C is a charac	ter string.
Other library mod	lules:			
Execution Time (micros	seconds): if loif length(C	ength(C) =0: ) > 0: 135.8	26.6 + 1.0 (if length	(c) > 4)
Input Arguments:  Type bit string	Precision_		How Passed R4	Units
Output Results:  Type  character string	Precision		How Passed R2 → descriptor	<u>Units</u>
Errors Detected: Error #		<u>Cause</u>		ixup
	etected in CSSI			
Comments: If the legiven, and nothing	ngth of the ing	out character	string is 0, no ange the length	error is of the

input string. Registers Unsafe Across Call: R5,F0,F1. The first character is set to 1 by clearing R5, the character width is Algorithm: set to the current length of the string. Processing continues as at

A in the description of the algorithm at entry CSSTP.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

_	CTT	٠.
٠.	514	,

Secondary Entry Name: CSSTP

Function: To store a bit string into a partitioned character string.

# Invoked by:

- X Compiler emitted code for HAL/S construct of the form: SUBBIT (CHAR\_STRING<sub>A TO B</sub>) = BIT\_STRING
- Other library modules:

Execution Time (microseconds): 148 + KA + KB, where KA = 1.0 if B-A>40 otherwise KB = 9.2 if A is even

0 otherwise

# Input Arguments:

Type	Precision	How Passed	Units
bit string		PA	-
integer(first ch	aracter) SP	R5	<b>-</b>
integer (last cha Output Results:	racter) SP	<b>R6</b>	
Type	Precision	How Passed	Units
character string		R2 → descriptor	

#### Errors Detected:

Error # Fixup character subscript out of legal range. (See CSLDP), except if last char < first char, then leave input

Comments: Registers Unsafe Across Call: R5,F0,F1. string un CSSTP cannot change the current length of the character string: string unchanged. gives error is subscript out of current legal range, Algorithm:

The character partition is checked for validity, and ERROR 17 is sent if anything is bad.

The character partition width is checked, If it is < 0, then the input character string is returned unchanged, If > 4, then it is set to 4. The first bit and last bit are determined as:

First bit = 1 + 8\*(first character-1) Last bit = First bit + 8\*character width-1

The first bit, last bit, and character width of the string are then sent to B under entry CPSST.

acto	
CSID	

Secondary Entry Name: CPSST

Function: To store a bit string into specified bits of a character string.

Invoked by:

Compiler emitted code for HAL/S construct of the form:
SUBBITA TO B (CHAR\_STRING) = BIT\_STRING

Other library modules:

Execution Time (microseconds): 114.4

Input Arguments:

<u>Type</u>	Precision How Passed	Units
bit string	R4	-
integer(first bit)		
integer(last bit)	kata (SP) a kata kata kata kata kata kata kata k	_
Output Results:		
Type	Precision How Passed	Units
character string	- R2 → descriptor	-

Errors Detected:

Error #			Cause			Fixun
30	subbit	partition	out of	legal	range.	Fixup (See CPSLD)

Comments: CPSST cannot change the current length of the input character string. In particular, a null character string input will result in a null string output. Registers Unsafe Across Call: R5,F0,F1.

Algorithm: Set character partition width to the current length of the character string. If it is 0, exit immediately after sending ERROR 30.

B Test subbit partition for validity, and send ERROR 30 if anything is bad. Find the first halfword containing the specified partition. The first bit relative to that halfword, and the bit partition width thus:

bit width = last bit - first bit + 1
first halfword = 1 + (first bit - 1)/16
relative first bit = first bit - 16(first halfword - 1)

Load 4 halfwords, beginning with the first halfword of the partition, into register pair R4-R5.

Prepare a mask with 0's in the specified bit positions and 1's else where as the 1's complement of:

(2bit width -1) (264-relative last bit)

where relative last bit = relative first bit + bit width - 1.

Use this mask to mask out the old bits in R4-R5. Shift the input bit string left by (64-relative last bit) positions to align it with the specified bit positions. Then OR it into the contents of R4-R5. Store this back into the character string, and exit.

CS	ĽD		

Secondary Entry Name: CPSSTP		er en et en transk en en Statistisk en
Function: To store a bit string into character string.	o selected bits of a pa	urtitioned
Invoked by:		
Compiler emitted code for HAL/S of SUBBIT A TO B (CHAR_STRING C TO D)		
Other library modules:		
Execution Time (microseconds): 145.0	+ 9.2(if C is even)	
Input Arguments:		
Type Precision integer (C) SP integer (D) SP (Continued on bottom of this page)	How Passed R5 R6	<u>Units</u>
Output Results:		
Type Precision character string	How Passed R2 → descrip	QT- Coulty ith reserve
Errors Detected:		
Error # Caracter subscript ou 30 subbit partition out o		Fixup (See CSSTP) (See CPSST)
Comments: CPSSTP cannot change the comments: Registers Unsafe Ac	rurrent length of the iross Call: R5,F0,F1.	nput character
Algorithm: Check character partition and exit if last char < first char on to 1 halfword before the first halfwo first and last bits by 8 if the first 8 bits of the halfword) after checkin error 30 if it is < 1. Then continue	currlen = 0. Reset of rd of the specified particles character is even (so g validity of first bi	character pointer artition, bumping o lies in low-order
Input Arguments (Continued)  Type Precision integer(A  B) (SP  SP)	How Passed R7	<u>Units</u>

REPRODUCIBILITY OF THE

		CSTR	UC	
HAL,	/s-FC LIBRARY ROU	TIME DESCRIPTION		
Source Member Name:	CSTRUC si	ze of Code Area:	12	Hw
Stack Requirement:	O Hw Da	ta CSECT Size:	0	Hw
X Intrinsi		<pre>Procedure</pre>		
Other Library Modules	Referenced: None	e.		
ENTRY POINT DESCRIPTION	ons			
Primary Entry Name: _	CSTRUC			
Function: Compares	two structures a	nd returns result (=	or ≠) in co	ndition
code.				
Invoked by:				
THYOUGH DY:				
X Compiler emitted		construct of the form	1;	
X Compiler emitted		construct of the form		
X Compiler emitted  If S1 = S2, THEN.	, where Sl and			
<pre>X Compiler emitted If Sl = S2,THEN.</pre> Other Library Mo	, where Sl and	S2 are structures.		
X Compiler emitted  If Sl = S2, THEN.  Other Library Mo	, where Sl and	S2 are structures.		red.
X Compiler emitted  If S1 = S2, THEN.	, where Sl and	S2 are structures.		red.
<pre>X Compiler emitted If Sl = S2,THEN.</pre> Other Library Mo	, where Sl and	S2 are structures.  10.4n, n = # halfw	words compar	
<pre>X Compiler emitted     If S1 = S2,THEN.  Other Library Mo  Execution Time (micro Input Arguments:     Type</pre>	, where Sl and	S2 are structures.  10.4n, n = # halfw	words compar	
<pre>X Compiler emitted     If S1 = S2, THEN.  Other Library Mo  Execution Time (micro  Input Arguments:</pre>	, where S1 and dules: eseconds): 5.4 +	S2 are structures.  10.4n, n = # halfw	words compar	
<pre>X Compiler emitted     If S1 = S2, THEN.  Other Library Mo  Execution Time (micro Input Arguments:     Type</pre>	, where Sl and dules: eseconds): 5.4 +  Precision	10.4n, n = # halfw  How Passed R2 + 1st HW	words compar	ced. Uni
<pre>X Compiler emitted     If S1 = S2,THEN.  Other Library Mo  Execution Time (micro  Input Arguments:</pre>	, where Sl and dules: eseconds): 5.4 +  Precision	10.4n, $n = \#$ halfw $\frac{\text{How Passed}}{\text{R2} \rightarrow \text{lst HW}}$ R3 $\rightarrow$ lst HW R5	words compar	<u>Un</u>
X Compiler emitted If S1 = S2, THEN. Other Library Mo  Execution Time (micro  Input Arguments:  Type Structure Structure Integer (count)  Output Results: Type	, where Sl and dules: eseconds): 5.4 +  Precision	10.4n, n = # halfw  How Passed R2 + lst HW R3 + lst HW R5	vords compar	<u>Un:</u>
If S1 = S2, THEN.  Other Library Mo  Execution Time (micro  Input Arguments:  Type Structure Structure Integer (count)  Output Results:	, where Sl and dules: seconds): 5.4 +  Precision SP	10.4n, $n = \#$ halfw $\frac{\text{How Passed}}{\text{R2} \rightarrow \text{lst HW}}$ R3 $\rightarrow$ lst HW R5	vords compar	<u>Uni</u> HW
If S1 = S2, THEN.  Other Library Mo  Execution Time (micro  Input Arguments:  Type Structure Structure Integer (count)  Output Results:  Type	, where Sl and dules: seconds): 5.4 +  Precision SP	10.4n, n = # halfw  How Passed R2 + lst HW R3 + lst HW R5	vords compar	<u>Uni</u>

Comments: In order that the correct code is in the C.C. on exit, it is reset immediately before branching back to the calling program (BCRE does not set the C.C.). An exclusive OR of a register with itself sets to C.C. to '00'(=). An OR of a non-zero register (R4 is used because, as the return address register, it is always assumed to be non-zero) resets to C.C. to '11'(\(\frac{1}{2}\)).

Registers Unsafe Across Call: R2,R3,R4,R5,R6.
Algorithm: The two structures are compared halfword-by-haflword until a pair does not match, or all of the halfwords are compared and found to be equal. The condition code is set by the compare instruction.

		CTOB		
HAL/S-FC LIBR	ARY ROUTINE DE	SCRIPTION		
Source Member Name: CTOB	Size of C	ode Area:	32	Hw
Stack Requirement: 18	Hw Data CSEC	T Size:	2	Hw
☐ Intrinsic	Pro	cedure		
Other Library Modules Reference	ed: GTBY	PE		
ENTRY POINT DESCRIPTIONS				
Primary Entry Name: CT				
Function: Conversion from chara	acter string d	ata to bit stri	.ng.	
Invoked by:	113T /C canalana	ot of the form		
Compiler emitted code for	character stri	ng.		
BIT <sub>@BIN</sub> (C), where C is a				
Other Library Modules:				
	NCHAR			
Execution Time (microseconds):	the state of the s	27.8 + KA, + KI	3 + KC)	
where: (Continued on bo	ttom k=1			
Input Arguments:		Bases of the second		Units
Type Precisi Character	<u>on</u>	How Passed R2 → descri		<u></u>
Clidiacter				
Output Results:		How Passed		Units
Type Precisi Bit length=31 in		R5		-
Errors Detected:	Cause		Fixup	
Error # 29 Input strin	ng not in stand	lard format.	Return	zero bit
			string.	
Comments: If the input stronger bits will be	ring includes I	more than 32 di	gits, thenses an err	n high- or.
Registers Unsafe				
Algorithm: Characters are	examined one by	y one. Blanks	are ignor	ed.
When a 'l' is encountered, a of the result register. When	'l' bit is shi a '0' is enco	fted into the I untered, a '0'	.ow-order	DIC
into the low-order bit of the				
into the low-order bit of the	NCHAR = leng	th (C)		4.4 if C\$(K)=h
into the low-order bit of the Execution Time (Continued):	NCHAR = length $KA_v = 1.2$	th(C) if X is odd X is even		4.4 if C\$(K)=b O otherwise.

	may bre biblond noor	NE DESCRIPTION	
Source Member Nam	ne: Size	of Code Area:	287 Hw
Stack Requirement	: 30 Hw Data	CSECT Size: 2	Hw
☐ Intr	rinsic	Procedure	
Other Library Mod	ules Referenced: GT	RYTE	
ENTRY POINT DESCR	TPTIONS		
Primary Entry Nam	ne: CTOE		
Function: Perfo	rms internal character	to single precision scalar	
	tted code for HAL/S coracter string>).	struct of the form:	
X Other Librar			
CSH	APQ	nd of algorithm descripti	
CSH Execution Time (m	APQ	nd of algorithm description	on.
CSH Execution Time (m Input Arguments:	APQ nicroseconds): See e		
CSH Execution Time (m Input Arguments: Type	APQ	How Passed	on. <u>Uni</u>
CSH Execution Time (m Input Arguments:	APQ nicroseconds): See e		
CSH Execution Time (m Input Arguments: Type	APQ nicroseconds): See e	How Passed	
CSH Execution Time (m Input Arguments:	APQ nicroseconds): See e	How Passed	
CSH  Execution Time (m  Input Arguments:  Type  character  Output Results:	APQ nicroseconds): See e <u>Precision</u>	How Passed R2 → descriptor	<u>Uni</u>
CSH Execution Time (m Input Arguments:	APQ sicroseconds): See e  Precision - Precision	How Passed R2 → descriptor  How Passed	<u>Uni</u>
CSH Execution Time (m Input Arguments:	APQ nicroseconds): See e  Precision  Precision  SP  Caus	How Passed R2 → descriptor  How Passed F0  Fix	Uni
CSH  Execution Time (m  Input Arguments:	APQ nicroseconds): See e  Precision  Precision SP	How Passed R2 → descriptor  How Passed F0  Fix	Uni
CSH  Execution Time (m  Input Arguments:	APQ nicroseconds): See e  Precision  Precision  SP  Caus	How Passed R2 → descriptor  How Passed F0  Fix	Uni
CSH  Execution Time (m  Input Arguments:	APQ nicroseconds): See e  Precision  Precision  SP  Caus	How Passed R2 → descriptor  How Passed F0  e Fixestandard format. Ref	Uni - - - ip

follows:

First, we set  $S_0 = 0$ . Now, at the k-th step,  $k \ge 1$ , we let  $S_k = 10 \star S_{k-1} + d_k$ , where  $d_k$  is the k-th digit in the string. All calculations are double scalar. When  $S_k$  becomes  $\geq$  X'4E19999A', all further digits are insignificant and are scanned for validity but otherwise ignored.

(Continued on next page)

5-299

CTOE

Algorithm (Con't)

This yields a scalar which may be incorrect by a power-of-10 multiple, but otherwise represents the decimal number of the left of the exponent. As for the power of 10, if a decimal point is encountered while scanning the input string, a count is kept of how many digits there are to the right of the decimal point in the input string. The negative of this count is stored in temporary location COUNTE for later use.

Next, the type of exponent (if any) is determined, and the value is calculated with a simple fixed-point calculation (ab 10 = 10a + b) and added to COUNTH, COUNTB, or COUNTE accordingly as the type of exponent is hexadecimal, binary, or decimal. Continue this process as long as there are remaining exponents.

If the end of the string is reached with no invalid characters found, then the scalar is modified according to the exponents already computed. First, the power-of-2 exponents are combined as 4H + B, since

$$16^{H} \cdot 2^{B} = 2^{4H + B}$$
.

The high part of this (power of 16) is added to the exponent of the scalar, while the low 2 bits control a loop in which the scalar is doubled 0-3 times.

Next, the decimal exponent, which has been combined with the correction for the decimal point in the input, is used as a power of 10 in the standard way of taking integral powers. The scalar intermediate is multiplied or divided by this result according to the sign of the exponent, completing the conversion. CTOE

# Timing:

```
88.4 + 11.0 • (floor (# leading blanks/2))
     + 12.0 (if # leading blanks odd)
     + 10.2 (# trailing blanks)
     + 2.0 (if + sign)
     + 7.0 (if - sign)
     + 59.6 (# significant digits where S_k < X'4E19999A')
     + 17.6 (# significant digits)
     + 47.2 (if at least 1 significant digit)
     + 62.4 (# significant digits where Sk > X'4E19999A')
     + 20.6 (if decimal point)
     + 9.6 (if no exponents of any kind)
     + 40.2 (if any exponents)
     + 9.6 (# E type exponents)
     + 15.2 (# H type exponents)
     + 18.2 · (# B type exponents)
     + 9.8 · (# exponents)
     + 37.8 · (# additional exponents)
     + 0.2 (# exponents with '+' sign)
     + 7.8 (# exponents with '-' sign)
     + 24.6 (total number of exponent digits)
     + 22.8 (if any B or H exponent)
     + 7.6 (total B exponent mod 4)
     + 14.0  (if p=0)
     +(17.8 + 27.8 div(p, 23)) (if p positive)
     +(18.8 + 28.8 div( p , 23)) (if p negative)
     + 23.2 ((# significant zeroes in the binary representation
                                                                          (if p \neq 0)
                 of |p| mod 23) - 1 (if |p| mod 23 is even)
     + 36.2 ((# significant ones in the binary representation
                 of |p| \mod 23) - 1(if |p| \mod 23 is odd)
     + 14.2 (if | p | mod 23 \neq 0)
     + 28.0
     + 1.6 (if p < 0)
     + 14.4 · ((# of even indexed characters after leading blanks(+1)
     + 15.6 ((# odd indexed characters after leading blanks)+1)
```

where p = total of E type exponents - (# significant digits after decimal point).

CTOE

Secondary Entry Name:

Performs internal character to double precision scalar Function:

conversion.

Invoked by:

X Compiler emitted code for HAL/S construct of the form:

SCALAR @DOUBLE (<character string>).

X Other library modules: **CSHAPQ** 

Execution Time (microseconds): Time is computed by CTOE formula - 1.8.

Input Arguments:

Precision Type

**How Passed** R2 → descriptor Units

Output Results:

Type Scalar

character

Precision

How Passed FO, F1

Units

Errors Detected:

Error #

Cause

Fixup

20

Input string not in standard format.

Return 0.0

Comments:

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Same routine as CTOE; all conversions result in a double Algorithm:

precision value of which the portion in Fl is discarded when single precision is desired by the caller of this routine.

OHT TO YELLOWING

		_	CTOI	
на	L/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	CTOI	Size of Code Area:	104 Hw	
Stack Requirement: _	20 Hw	Data CSECT Size:	2 Hw	
☐ Intrins	ic	X Procedure		
Other Library Module	s Referenced:	GTBYTE		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	CTOI		ing jarah salah	
		ring to a double preci	sion integer.	
Invoked by:				
Compiler emitte	ed code for HA	L/S construct of the f	orm:	
INTEGER @DOUBLE	( <character st<="" td=""><td>tring&gt;) or BIT<sub>@DEC</sub>(<cha< td=""><td>Idorer serans /</td><td></td></cha<></td></character>	tring>) or BIT <sub>@DEC</sub> ( <cha< td=""><td>Idorer serans /</td><td></td></cha<>	Idorer serans /	
X Other Library M	Modules: cs	SHAPO		
Execution Time (micr	roseconds): (c	ica mart maga)		
Everage true ("To-	(5	see hext pages.		
Input Arguments:				
Type	Precision	How Pas	<u>sed</u> , <u>Un</u>	nits
Character		R2 → desc	riptor	
Output Results: Type	Precision	How Pas	sed Ur	nits
Integer	DP	R5		-
Errors Detected:				
· Error #		Cause	<u>Fixup</u>	
Iı	nput string no	t in standard format.	Return 0	
Comments:  Registers U	nsafe Across C	Call: R5,F0,F1.		
			input string. If a	
Algorithm: First, 1	eading blanks	are stripped from the ag is set. The basic of	onversion proceeds	46
minua gian is enco			en la la companya di	
as follows: Initia	lize in to 0.	At step K, K / I, Iel	TK - TO TK-T OK	
as follows: Initia	lize i <sub>0</sub> to 0. th digit in th	At step k, $k \ge 1$ , let ne input string. At thord integer. This sign	e end, this fixed-	

the result is shifted left 16 bits if a halfword answer is required.

1

CTOI

# Exeuction Time (microseconds):

where k = 72.6

```
k + 11.0 (floor (# leading blanks/2))
+ 18.6 (if # leading blanks odd)
+ 9.4 (if '-' sign)
+ 10.6 (if first character is a number)
+ 15.6 (# even index characters after leading blanks)
+ 14.4 (# odd index characters after leading blanks)
+ 13.0 (if # trailing blanks > 0)
+ 8.4 (# trailing blanks)
+ 28.2 ((# non blank characters) -1)
```

1

CTOI	

Secondary Entry Name:	: CTOH		
Function: Converts a c	haracter string to a	single precision integer.	
Invoked by:			
Compiler emitted INTEGER @SINGLE	d code for HAL/S cons ( <character string="">)</character>	struct of the form:	
v Other library m	nodules: CSHAPQ		
Execution Time (micr	coseconds): Same as	for CTOI, except $k = 74.4$ .	
Input Arguments: Type Character	Precision	How Passed R2 → descriptor	<u>Units</u>
Output Results: Type Integer	Precision SP	How Passed R5	Units
Errors Detected:	<u>Cau</u>		
22 Inpu	t string not in stand	ard format. Return 0	

Comments:

Registers Unsafe Across Call: R5,F0,F1.

Algorithm:

See CTOI.

Secondary Entry Name: CTOK
Function: Converts a character string to a 32-bit string for use with the @DEC of the BIT conversion function.
Invoked by:
X Compiler emitted code for HAL/S construct of the form: BIT@DEC ( <character string="">).</character>
Other library modules:
Execution Time (microseconds): (See below).
Input Arguments:  Type Precision How Passed United the R2 + descriptor
Output Results:  Type Precision How Passed Uni
bit string 32-bits R5
Errors Detected:  Error # Cause Fixup  Return 0
Comments: Registers Unsafe Across Call: R5,F0,F1.
Algorithm: See CTOI.
Execution Time: 85.8 + 11.0 (floor (# leading blanks/2)) + 18.6 (if # leading blanks odd) + 15.6 (# even index characters after leading blanks) + 14.4 (# odd index characters after leading blanks) + 13.0 (if # trailing blanks > 0)

REPRODUCIBILITY OF THE ORIGINAL PAGE 15 FOOR

+ 28.2 (# non blank characters - 1)

			TOX	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name	: CTOX	Size of Code Area:	58	Hw
Stack Requirement:	18 Hw	Data CSECT Size:	4 1	Hw
☐ Intri Other Library Modu		Procedure GTBYTE		
ENTRY POINT DESCRI	PTIONS			
Primary Entry Name	: CTOX			
Function: Convers	sion from charact radix.	er string to bit strin	g, hexadecimal	
	ted code for HAL, where C is a cha	/S construct of the for	) <b>::::::</b> ;	
Other Library	Modules:			
Execution Time (mi	croseconds): (	See next page).		
Input Arguments: Type	Precision	How Pass	se <b>d</b>	Units
Character		R2 → descrip	tor	_
Output Results:				
<u>Type</u>	Precision	How Pass	sed.	Units
Bit	32 bits			***
Errors Detected: Error #		<u>Cause</u>	<u>Fixup</u>	
32		tandard hexadecimal	Return 0	

Comments: Imbedded blanks, or leading or trailing blanks, are all considered invalid characters. An input string too long to be accommodated in 32 bits will cause high order bits to be lost. Registers Unsafe Across Call: R5,F0,F1. Algorithm: Characters are fetched one by one. In the CTOX entry,

Algorithm: Characters are fetched one by one. In the CTOX entry, characters 'A' - 'F' are converted to their bit equivalents, characters '0' - '9' are passed on to the common section, and an error is signalled if the input character lies between X'39' and X'41' (DEU characters '9' and 'A' respectively). In the CTOO entry, an error is sent if the character is greater than X'37' (DEU character '7'). Other characters are passed to the common sechon for translation.

In the common section, decimal digits 0-9 (0-7 for octal) are translated to their bit equivalents, and an error is sent if the character precedes '0' in the collating sequence. These bit equivalents, and the ones passed from the CTOX section, are shifted into the low-order 4 bits (3 for octal) of the result register.

CTOX

Execution Time (microseconds):

32.0 + 
$$\sum_{k=1}^{NCHAR}$$
 (33.6 +  $KA_k$  +  $KB_k$ )

where:

NCHAR = length(C)

 $KA_X = 0$  if C\$(X) is alphabetic 6.8 if C\$(X) is numeric

KB<sub>X</sub> = 1.2 if X is odd 0 if X is even

CTOX		

Secondary Entry Name: CTOO

Function: Conversion from character string to bit string, octal radix.

Invoked by:

Compiler emitted code for HAL/S construct of the form:

BIT @OCT (C), where C is a character string.

Other library modules:

NCHAR

Execution Time (microseconds): 33.4 +  $\sum_{k=1}^{K}$  (34.2 +  $KA_k$ ), where

NCHAR = length(C),

KA<sub>X</sub> = 1.2 if X is odd 0 otherwise

Input Arguments:

<u>Type</u> Character Precision

How Passed R2 → descriptor Units

Output Results:

Type Bit Precision 32-bits

How Passed R5 Units

Errors Detected:

Error #

String not in standard octal conversion format.

**Fixup** Return 0

Comments:

Registers Unsafe Across Call: R5,F0,F1.

Algorithm: See CTOX.

			DSLD	
	HAL,	/S-FC LIBRARY ROU	TINE DESCRIPTION	
	Source Member Name: _	DSLD S:	ize of Code Area: 22	Hw Hw
	Stack Requirement:	18 Hw D	ata CSECT Size:	2 Hw
	☐ Intrinsi		XX Procedure	
	Other Library Modules	Referenced:	None,	
	ENTRY POINT DESCRIPTI	ONS		
	Primary Entry Name: _	DSLD		
			a double precision scalar	
\$ 7				
	Invoked by:			
	X Compiler emitted	code for HAL/S	construct of the form:	
			FIRST TO LAST (SCALAR VA	R <sub>a DOUBLE</sub> );
			11104 10 11101	
	Other Library Mo			
	Other Library Mo	dules:		
	Execution Time (micro	dules:		
	Execution Time (micro	dules: oseconds): 36.5	How Passed	Units
	Execution Time (micro Input Arguments: Type Scalar	odules: oseconds): 36.5  Precision DP	How Passed R2 → Scalar	
	Execution Time (micro Input Arguments: Type Scalar Integer(first bit	eseconds): 36.5  Precision DP SP	How Passed R2 → Scalar R5	
	Execution Time (micro Input Arguments:  Type Scalar Integer(first bit Integer(last bit)	odules: oseconds): 36.5  Precision DP	How Passed R2 → Scalar	
	Execution Time (micro Input Arguments:  Type Scalar Integer(first bit Integer(last bit) Output Results:	eseconds): 36.5  Precision DP SP	How Passed R2 → Scalar R5	
	Execution Time (micro Input Arguments:  Type Scalar Integer(first bit Integer(last bit)	odules: oseconds): 36.5  Precision DP SP SP	How Passed R2 → Scalar R5 R6	Units
	Execution Time (micro Input Arguments:	Precision DP SP SP Precision	How Passed R2 → Scalar R5 R6 How Passed	Units
	Execution Time (micro Input Arguments:	Precision DP SP SP Precision Area of the precision of the	How Passed R2 → Scalar R5 R6 How Passed R5	Units Units
	Execution Time (micro Input Arguments:	Precision DP SP SP Precision Area of the precision of the	How Passed R2 → Scalar R5 R6  How Passed R5  Cause out of range. 1) If	Units  Units  Fixup First bit < 1, set t
	Execution Time (micro Input Arguments:	Precision DP SP SP Precision Area of the precision of the	How Passed R2 → Scalar R5 R6  How Passed R5  Cause out of range. 1) If	Units Units
	Execution Time (micro Input Arguments:	Precision DP SP SP Precision Area of the precision of the	How Passed R2 → Scalar R5 R6  How Passed R5  Cause out of range. 1) If	Units  Units  Fixup First bit < 1, set t
	Input Arguments:  Type Scalar Integer(first bit Integer(last bit) Output Results: Type Bit string  Errors Detected: Error # 30  Comments:	Precision DP SP SP Precision Area of the precision of the	How Passed R2 → Scalar R5 R6  How Passed R5  Cause out of range. 1) If	Units  Units  Fixup First bit < 1, set t

Algorithm: Get the double word operand in register pair R2-R3. If first bit -1 < 0, then given ERROR 30 and set to 0. Use first bit 1 as left shift count to eliminate unwanted high order bits. Compute 64 - last bit + first bit -1, and give ERROR 30 and set to 0 if it is < 0. Use this as right shift count to justify bit string in R3. Return contents of R3.

REPRODUCIBILITY OF THE ORIGINAL PAGE 15 POOR

			DSST	
		UNI /C. EC TIDDADY DOMINIE I	DECORTORION	
*.		HAL/S-FC LIBRARY ROUTINE I	DESCRIPTION	
1	Source Member Name	: DSST Size of	Code Area: 54	Hw
	Stack Requirement:	18Hw Data CSI	ECT Size:	2 Hw
	☐ Intri	nsic $ abla$ Pr	ocedure	
	Other Library Modu	les Referenced: None		
	ENTRY POINT DESCRI	PTIONS		
	Primary Entry Name	: DSST		
	Function: Store	es a bit string into selec	ted bits of a double wo	rd
	Invoked by:  X Compiler emit	ted code for HAL/S constru	uct of the form:	
	SUBBIT A TO I	GOUBLE_SCALAR) = BIT_STR	ING	
	Other Library	Modules:		
	Execution Time (mi	croseconds): 64.6		
	Input Arguments:	성인도 <u>원</u> 경 이 유명 등 등 기계 등 하다.		
	Type	Precision	How Passed	Units
	Integer (A)	SP Area (a. a. a	R5 R6	
	Integer (B)	SP 18 8 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	R7	
	Bit string Output Results:	요 이 항상 폭제 길이 보는 방법 회육 소		
	Type	Precision	How Passed	Units
	Scalar	DP	R2 → scalar	
	Errors Detected:			
Kj.	Error #	<u>Cause</u>	<u>Fixup</u>	
	30	Subbit partition illega	<ol> <li>1) If first bit</li> <li>2) If last bit &gt;</li> </ol>	
	Comments:			
		e Across Call: None.		
	Algorithm: Check last bit, and go a mask of width	first bit. If < 1, send et the partition width as = partition width as 2wid	last bit - first bit + $^{\text{th}}$ - 1. If last bit $\leq$	l. Create 64, shift

Algorithm: Check first bit. If < 1, send error 30 and set to 1. Save the last bit, and get the partition width as last bit - first bit + 1. Create a mask of width = partition width as 2width - 1. If last bit < 64, shift left by 64 - last bit. If last bit > 64, send error 30 and set last bit to 64 by shifting right by last bit - 64. Then invert the (doubleword) mask. Mask out the selected bits of the operand in storage. Then, shift the input bit string to the right position (left 64 - last bit, or right last bit - 64), and OR to the operand in storage, completing the operation.

			ETOC	
HAL/S	S-FC LIBRARY	ROUTINE DESCRIPTION	1	
	ETOC	Size of Code Area:	070	Hw
Stack Requirement:	20 Hw	Data CSECT Size:	64	Hw
☐ Intrinsic		Procedure		
Other Library Modules	Referenced:	None		
 ENTRY POINT DESCRIPTION	NS			
Primary Entry Name:	ETOC	· · · · · · · · · · · · · · · · · · ·		
Function: Converts a character-st	single pred	cision scalar to sta for a scalar.	ndard internal	
Invoked by:				
CHARACTER (SCAL		L/S construct of the	form:	
Compiler emitted	AR_VAR)	L/S construct of the	form;	
CHARACTER (SCAL	AR_VAR)		form;	
CHARACTER (SCAL	AR_VAR)	336.9	Passed_	Units
CHARACTER (SCAL  Other Library Mod  Execution Time (micros  Input Arguments:  Type  Scalar  Output Results:  Type	Precision  Precision	336.9  How F	Passed	Units
CHARACTER (SCAL  Other Library Mod  Execution Time (micros  Input Arguments:  Type  Scalar  Output Results:	Precision	336.9  How F	Passed)	

maxlength of output string is ignored. Comments: Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Algorithm:

Clear Fl to convert to double precision. Determine the sign and get the absolute value of the input. If input = 0, output string is '80.0' padded with blanks to length 14 (length 23 for DTOC). The next operation reduces the exponent of the scalar to X'41' keeping track of the change in exponent that this requires. Since  $\log_{10}x = (\log_{10}16)(\log_{16}x)$ , this is done by getting (exponent - 65)\* $\log_{10}16$ and using this as an exponent of 10, dividing the scalar by the result. It is possible for this to be off by 1, so another pass is made before continuing. At this point, the number is between 1 and 16. If it is greater than or equal to 10, multiply by 1/10 and record the exponent as one greater,

(Continued on next page)

ETOC

Algorithm (Con't)

This causes the first decimal digit of the number to be the first hexadecimal digit of the scalar, in bits 8-11 of F0. This is stored, together with a blank if the value is  $\geq 0$ , or a '-' if the value < 0. The remaining mantissa is in fractional form in F0-F1. This hexadecimal fraction is converted to decimal digit-by-digit by successive multiplication by 10. One digit is generated and stored with the decimal point, then 6 digits are stored in the next three halfwords.

The sign of the exponent (as calculated above) is tested, and either 'E+' or 'E-' is stored in the next halfword. Two decimal digits of exponent are stored in the last halfword.

ETOC		

Function: Converts character-string fo		calar to standard internal	
Invoked by:			
x Compiler emitted	code for HAL/S const	cruct of the form:	
CHARACTER (DOU	BLE_SCALAR)		
Other library mo	dules:		
Execution Time (micro  Input Arguments:  Type	seconds): 602.5  Precision	How Passed	Units
Scalar	<b>DP</b>	FO-F1	
Output Results: Type Character string	Precision length=23	<pre>How Passed R2 → descriptor</pre>	<u>Units</u>
Errors Detected: Error #			
	Cause	Fixup	rings of the second

Comments:

Registers Unsafe Across Call: F0,F1,F2,F3,F4,F5.

Also, rather than 6 digits being stored in the loop, 14 are computed and stored. The exponent section also looks different, as one more digit is stored with the exponent, changing its alignment, thus storing successively '<digit>E', '\*digit>I', '<digit>Garbage>' in the last 3 halfwords.

RETECTION OF THE CONTROL OF THE CONT

	AL/S-FC LIBRARY RO	OUTINE DESCRIPTION	
Source Member Name:	131011		14
Stack Requirement:	0 Hw 1	Data CSECT Size: 0	1
🛚 Intrin	sic Sector 0	<pre>Procedure</pre>	
Other Library Modul	es Referenced:	None	
ENTRY POINT DESCRIP	TIONS		
Primary Entry Name:	ЕТОН		
Function: Converts integer.	single precision	scalar value to single prec	ision
Invoked by:  X Compiler emitt	ed code for HAL/S	construct of the form:	
X Compiler emitt	e I is a single pr S is a single pr Modules: QSHAN	ecision integer, and ecision scalar.	
X Compiler emitt I = S; wher	e I is a single pr S is a single pr Modules: QSHAN	ecision integer, and ecision scalar.	
X Compiler emitt  I = S; wher  Other Library  Execution Time (mic	e I is a single pr S is a single pr Modules: QSHAN	ecision integer, and ecision scalar.	
X Compiler emitt I = S; wher	e I is a single pr S is a single pr Modules: QSHAN	ecision integer, and ecision scalar.	
Compiler emitt  I = S; wher  Other Library  Execution Time (mic	e I is a single pr S is a single pr Modules: QSHAN	ecision integer, and ecision scalar.	
X Compiler emitt  I = S; wher  Other Library  Execution Time (mic  Input Arguments:  Type  Scalar	e I is a single pr S is a single pr Modules: QSHAN Proseconds): 15.	ecision integer, and ecision scalar. PQ .4  How Passed	
X Compiler emitt  I = S; wher  Other Library  Execution Time (mic  Input Arguments:  Type  Scalar  Output Results:	e I is a single pr S is a single pr Modules: QSHAN Proseconds): 15.	ecision integer, and ecision scalar. PQ .4  How Passed	
Compiler emitt  I = S; wher  Other Library  Execution Time (mic  Input Arguments:  Type  Scalar	e I is a single pr S is a single pr Modules: QSHAN croseconds): 15,  Precision SP	ecision integer, and ecision scalar.  PQ  How Passed F0	
Compiler emitt  I = S; wher  Other Library  Execution Time (mic  Input Arguments:  Type Scalar  Output Results:  Type	e I is a single pr S is a single pr Modules: QSHAP croseconds): 15.  Precision SP  Precision	ecision integer, and ecision scalar.  PQ  How Passed  FO  How Passed	
<pre>X Compiler emitt I = S; wher  V Other Library  Execution Time (mic Input Arguments:</pre>	e I is a single pr S is a single pr Modules: QSHAP  croseconds): 15.  Precision SP  Precision SP	How Passed FO  How Passed R5	xup
Compiler emitt  I = S; wher  Other Library  Execution Time (mic  Input Arguments:  Type Scalar  Output Results:  Type Integer  Errors Detected:	e I is a single pr S is a single pr Modules: QSHAP  croseconds): 15.  Precision SP  Precision SP	How Passed F0  How Passed R5	<b>xup</b>

Algorithm: The six most significant hex digits of the scalar argument are converted to a fullword integer value. The 4 most significant hex digits of the integer value are left in the top halfword of the fixed point register after rounding the fraction.

ETOH	

Secondary Entry Name	: DTOH			
Function: Converts precision in		ion scalar value to	a single	
Invoked by:				
X Compiler emitte	d code for HAL/	S construct of the	form:	
I = D; where $I$	I is a single pr D is a double pr	ecision integer, an ecision scalar,	<b>a</b>	
Other library m	odules:			
Execution Time (micr	roseconds): 17.4			
Input Arguments: Type	Precision	How P	assed	Units
Scalar	.DP	FC	•	-
Output Results:  Type  Integer	Precision SP	How P	<del>dili di dia</del> dia 1964 any any a	Units -
Errors Detected: Error #		<u>Cause</u>	<u>Fixup</u>	
Comments: Registers Un	safe Across Cal.	L: R4,R5,F0,F1.		

Algorithm: See ETOH.

			GTBYTE	<u>-</u>
HAL/	S-FC LIBRARY	ROUTINE DESCRIPTION	ON	
Source Member Name:	GTBYTE	Size of Code Are	a: <u>14</u>	Hw
Stack Requirement:	0 Hw	Data CSECT Size:	0	Hw
🙀 Intrinsic		☐ Procedure		
Other Library Modules	Referenced:	None.		
ENTRY POINT DESCRIPTION	NS			
Primary Entry Name:	GTBYTE			
Function: Fetches one manipulation by o	character frother library	om a character str routines.	ing, Used for C	haracter
Invoked by:  Compiler emitted	code for HAI	L/S construct of th	ne form:	
Q Other Library Moo		CTOE, CLJSTV, CIN	DEX, CRISTV, CTO	e e
Other Library Mod	CTOI,	CTOX, CTRIMV.	byte	DB,
Execution Time (micros Input Arguments: Type	CTOI, seconds): 14 15 Precision	CTOX, CTRIMV.  1.4 to obtain lower 1.6 to obtain upper  How 1.72 *1HW in	byte byte Passed Front of HW to	Uni Eetch fro
Execution Time (microstant) Input Arguments: Type	CTOI, seconds): 14 15 Precision	CTOX, CTRIMV.  1.4 to obtain lower 1.6 to obtain upper  How R2 >1HW in lower half of R2	byte byte Passed Front of HW to	Unite Fetch from byte
Input Arguments:  Type pointer flag(which byte to Output Results: Type	CTOI, seconds): 14 15  Precision fetch) -  Precision	CTOX, CTRIMV.  1.4 to obtain lower 1.6 to obtain upper  How R2 >1HW in lower half of R2	byte byte  Passed front of HW to	Unit fetch from byte Unit

Algorithm: The halfword off of the pointer is loaded into a register for manipulation. If the flag indicates the upper byte is requested, the register is shifted right 8 bits and the lower half of the register is cleared to leave only the desired byte in the upper halfword of the register. If the flag indicates the lower byte is requested, then the first byte of the register is cleared. The flag is reset to indicate the upper byte if the lower byte was requested and vice versa, and the pointer is updated if the fetched byte was even. This is done now since GTBYTE is usually called a number of times from a loop.

		•		
			ITOC	
	HAL/S-FC LIE	RARY ROUTINE DE	SCRIPTION	
	Source Member Name: ITOC	Size of C	ode Area: 104	Hw
e filosofie	Stack Requirement: 28	Hw Data CSEC	T Size: 0	Hw
	<pre>Intrinsic</pre>	X Proc	cedure	
	Other Library Modules Reference	ed:	None.	
	ENTRY POINT DESCRIPTIONS			
	Primary Entry Name: ITOC			
	Function: Converts a fullwor string format for integers.	d integer into	standard internal char	acter
	Invoked by:  Compiler emitted code for	HAL/S construc	t of the form:	
	CHARACTER (FULLWORD_INTEG			
	Other Library Modules:  Execution Time (microseconds):	254.6		
	Present Time (microscolles).	254.0		
	Input Arguments:			
	Type Precisi	<u>on</u>	How Passed	Units
	Integer DP		R5	
	Output Results:			
	<u>Type</u> Precisi	on	How Passed	Units
	Character string -		R2 > descriptor	
	Errors Detected:			
¥	Error #	Cause	Fixup	
		s or leading or gisters Unsafe A	trailing blanks. Max Across Call: None.	length
	Algorithm: Digits are genera	ed one by one.	Thus: Let I = input	integer.

Algorithm: Digits are generated one by one. Thus: Let I = input integer. Then:

 $d_k = I_k - 10(I_k/10)$  integer multiply and divide.  $I_{k+1} = (I_k - d_k)/10$ .

The process terminates when  $I_k=0$ . As pairs of digits are generated, they are stored, right to left, in a temporary output area. The temporary result is then given a sign if necessary and moved to the output area. If an odd number of characters were generated, the move is with 8 bits offset for left alighment.

ITOC		

Secondary Entry Name:	HTOC	· 		
Function: Converts a string format for i	halfword int	eger into s	tandard internal char	acter-
Invoked by:				
CHARACTER (HALFW	code for HAL	/S construc	t of the form:	
Other library mod	dules:			
Execution Time (micros	seconds): 18	9.6		
Input Arguments:				
<u>Type</u> Integer	Precision SP		How Passed R5	Units
**************************************	SF		R5	
Output Results:				
<u>Type</u>	Precision		How Passed	Units
Character string			R2 → descriptor	
Errors Detected:				
· Error #		Cause	Fixu	2

Comments: No leading zeroes or leading or trailing blanks, Maxlength of output area ignored.

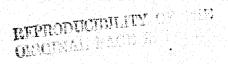
Registers Unsafe Across Call: None.

Algorithm:

Shift right algebraic 16 to convert single integer to double. Then proceed as in ITOC.

		ITC	D	
AL/S-FC LIBRARY	ROUTINE DESC	RIPTION		
ITOD	Size of Cod	e Area:	20	Hw
0 <sub>Hw</sub>	Data CSECT	Size:	0 .	Hw .
sic Sector O	☐ Proced	lure		
es Referenced:	None			
rions				
ITOD				
a <b>d</b> ouble precis	ion integer t	o a double p	recision	
od gode for HAT.	/s construct	of the form	•	
			•	
I is a double p	precision int	eger.		
Modules: OSHAT	PO			
<b>XD</b>	· ×			
roseconds): 15.	6			
Precision		How Passed		Uni
TICOIDION	e garanta ayan	11011 1 400004		
DP		R5		-
Precision		How Passed		Uni
DP		FO		_
	Cause		Fixup	
	<u>Cause</u>		Fixup	
	Cause		Fixup	
	Cause		Fixup	
			<u>Fixup</u>	
nsafe Across Ca		,F1.	Fixup	
nsafe Across Ca rst register of	ll: R4,R5,F0			
	O Hw sic Sector O es Referenced: FIONS  ITOD  a double precis ed code for HAL D is a double I is a double Hodules: QSHAL roseconds): 15.  Precision  DP	O Hw Data CSECT  sic Sector O Proced es Referenced: None  TIONS  ITOD  a double precision integer to  ed code for HAL/S construct D is a double precision scal I is a double precision inte  Modules: QSHAPQ  roseconds): 15.6  Precision  DP	O Hw Data CSECT Size:  Sic Sector O Procedure  ES Referenced:  None  PIONS  ITOD  A double precision integer to a double precision scalar, and I is a double precision integer.  Modules: QSHAPQ  Precision How Passed  DP R5	O Hw Data CSECT Size: O.  Sic Sector O Procedure  None  TTOD  a double precision integer to a double precision  ed code for HAL/S construct of the form:  D is a double precision scalar, and I is a double precision integer.  Modules: QSHAPQ  roseconds): 15.6  Precision How Passed  DP R5

value of the argument is loaded into the second register of the pair, and the result is normalized to a double precision scalar by adding D'O',



				ITOE	<b>-</b>
	HAL	/S-FC LIBRARY	ROUTINE DESCRIPTION	ON	
s	Source Member Name:	ITOE	Size of Code Area	a: 6	Hw
	Stack Requirement:	O Hw	Data CSECT Size:	0	Hw
•	▼ Intrinsi		Procedure		
C	Other Library Modules	Referenced:	None		
E	ENTRY POINT DESCRIPTI	ONS			
	Primary Entry Name: _	ITOE	The second secon		
F	Function: Converts a scalar.	double precis	sion integer to a	single precision	
<b>.</b>	Invoked by:    X   Compiler emitted   S = I; where S   I	is a single p	/S construct of th precision scalar, a precision integer.	ind	
	X Other Library Mo	odules: QSH1	<b>∆</b> PQ		
I	Execution Time (micro	seconds): 12.			
	Input Arguments:				
	Type	Precision	e bij gillikan, be geli littlikti	Passed	Units
	Integer	DP		R5	
	Output Results:				
	Type	Precision		Passed	Units
	Scalar	SP.		FO	
	Errors Detected: Error #		Cause	Fixup	
	Comments:	Unsafe Acros	s Call: R4,R5,F0,	<b>F1.</b>	
	Algorithm: The inte	eger argument	is converted to fl	loating point by	the CVFL

Algorithm: The integer argument is converted to floating point by the CVFL instruction and the binary point is adjusted by multiplication by a scale factor.

1				KTOC	
		/a ma tronany	POTIMITALE DESCRIPTION		
	HAL		ROUTINE DESCRIPTION	ranger (1986)	
	Source Member Name: _	KTOC	Size of Code Area:	70	Hw
	Stack Requirement:	0 Hw	Data CSECT Size: _	0	Hw
	☐ Intrinsi	.c	Procedure		
	Other Library Modules	Referenced:	NOTICE.		
	ENTRY POINT DESCRIPTI	ONS			raining and services. The services are services and services are services and services are services and services are services and services are services are services and services are servi
	Primary Entry Name: _	KTOC			
	Invoked by:		haracter conversion		<b>x.</b>
	Compiler emitted	d code for HAL	/S construct of the	form:	
	CHARACTER @DEC	(BIT_STRING)			
	Other Library Mo	odules:			
	Execution Time (micro	oseconds): 26	2.5 (for 16 bits)		
					Mary and the state of
	Input Arguments:				
	Input Arguments: Type	Precision	How Pa		Units
			R5		Units
	Type Bit string integer(length of bit string)	Precision SP			Units
	Type Bit string integer(length of bit string) Output Results:		R5 R6 <u>How P</u>	assed	Units - Units
	Type Bit string integer(length of bit string)	SP	R5 R6 <u>How P</u>		
	Type Bit string integer(length of bit string) Output Results: Type	SP	R5 R6 <u>How P</u>	assed	
	Type Bit string integer(length of bit string) Output Results: Type character string Errors Detected:	SP	$ \begin{array}{c} R5 \\ R6 \\ \underline{\text{How P}} \\ R2 + d \end{array} $	assed escriptor	
	Type  Bit string integer(length of bit string) Output Results:  Type character string  Errors Detected: Error #	SP Precision -	R5 R6  How Performance R2 $\rightarrow$ d  Cause  ea ignored. No lead	assed escriptor Fixup	Units
	Type  Bit string integer(length of bit string) Output Results:  Type character string  Errors Detected: Error #  Comments: Maxlengt "Sign bit" of inpure Registers Un	Precision  h of output ar ut string igno safe Across Ca	R5 R6  How Performance R2 $\rightarrow$ d  Cause  ea ignored. No lead	assed escriptor Fixup ling or trailing b	Units
	Type  Bit string integer(length of bit string) Output Results:  Type character string  Errors Detected: Error #  Comments: Maxlengt "Sign bit" of inpure Registers Un	Precision  h of output ar ut string igno safe Across Ca ength of the c	R5 R6 How Post Post Post Post Post Post Post Post	assed escriptor Fixup ling or trailing b R6,R7,F0,F1.	Units
	Type  Bit string integer(length of bit string) Output Results:  Type character string  Errors Detected: Error #  Comments: Maxlengt "Sign bit" of inpure Registers Un	Precision  h of output ar ut string igno safe Across Ca ength of the c	R5 R6  How Performed Property of the property	assed escriptor Fixup ling or trailing b R6,R7,F0,F1.	Units
	Type Bit string integer(length of bit string) Output Results: Type character string  Errors Detected: Error #  Comments: Maxlength "Sign bit" of inp Registers Un Algorithm: The 1	Precision  h of output ar ut string igno safe Across Ca ength of the c 1 + (log <sub>10</sub> <sup>2</sup> is computed fr	R5 R6  How Performed Property of the property	escriptor  Fixup  ling or trailing b  R6,R7,F0,F1.  computed as:	Units
	Type Bit string integer(length of bit string) Output Results: Type character string  Errors Detected: Error #  Comments: Maxlengt! "Sign bit" of inpotent of inpot	Precision  Precision  A contract of the contra	R5 R6 How Port R2 + d  Cause  ea ignored. No lead red.  11: R1,R2,R3,R4,R5, haracter string is continuous trum om this as:  ount = (1 + character one at a time, £rom	assed escriptor  Fixup  ling or trailing b  R6,R7,F0,F1. computed as: heated to an integ	Units
	Type Bit string integer(length of bit string) Output Results:  Type character string  Errors Detected: Error #  Comments: Maxlengt "Sign bit" of inp Registers Un Algorithm: The 1  A halfword count  Decimal digits thus: Let I 0 = in	Precision  Precision  h of output ar  ut string igno  safe Across Ca  ength of the co  1 + (log <sub>10</sub> <sup>2</sup> is computed fr  halfword co  are generated  put string as	R5 R6 How Port R2 + d  Cause  ea ignored. No lead red.  ll: R1,R2,R3,R4,R5, haracter string is continuous truncom this as:  ount = (1 + character)	assed escriptor  Fixup  ling or trailing b R6,R7,F0,F1. computed as: ncated to an integ count)/2 n right to left,	Units

KTOC

KTOC

Algorithm (Con't)

The process terminates when the halfword count is reached, with two digits stored per halfword. At the end, if an odd number of characters have been stored, the string must be shifted one character to the left for proper alignment.

	and the second			MSTRUC	<u>-</u>
	HAL	/S-FC LIBRARY	ROUTINE DESCRIPTION	NC	
Source Membe	er Name: _	MSTRUC	Size of Code Area	ı:8	Hw
Stack Requir	cement:	O Hw	Data CSECT Size:	0	Hw
	] Intrinsi	c	☐ Procedure		
Other Librar	y Modules	Referenced:	None		
ENTRY POINT	DESCRIPTION	ONS			
Primary Entr	y Name:	MSTRUC			
Function:	Moves a st	tructure.			
Trivole of here					
Invoked by:					
X Compile	r emitted	code for HAL	/S construct of the	form:	
S1 = S2	Sl and S2	are structur		. 101111	
31 - 32,	or and Sz	are structur	es.		
Othor I	ibrary Mod	3,,700.			
Ocher n	TDLALY MOC	vores:			
Evecution Ti	me (micros				
Execution Tir	me (micros	seconds): 4	.2 + 9.4n, n = # 1	nalfwords moved	
Execution Time	me (micros	seconds): 4	.2 + 9.4n,  n = # 1	nalfwords moved	
Execution Time		seconds): 4	.2 + 9.4n,  n = # 1	nalfwords moved	
		seconds): 4  Precision		nalfwords moved	
Input Argume	nts:		How P	assed	
Input Argumen	n <b>ts:</b> re (S2)		How P R2 → fir	assed st HW	<u>Uni</u>
Input Argumen <u>Type</u> Structur	n <b>ts:</b> re (S2)	Precision -	How P R2 → fir	assed	<u>Uni</u>
Input Argumen <u>Type</u> Structur	nts: re (S2) (#HW)	Precision -	How P R2 → fir	assed st HW	<u>Un:</u>
Input Argumen Type Structur Integer	nts: re (S2) (#HW)	Precision SP	How P R2 → fir R	assed st HW	<u>Un:</u> HV
Input Argumen Type Structur Integer	nts: re (S2) (#HW) ts:	Precision -	How P R2 → fir F	assed st HW 5	<u>Un:</u> HV
Input Argumen Type Structur Integer Output Resul	nts: re (S2) (#HW) ts:	Precision SP	How P R2 → fir R	assed st HW 5	<u>Un:</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F	assed st HW 5	<u>Un:</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F  How P R1 → fir	assed st HW 25 assed st HW	<u>Un</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F	assed st HW 5	<u>Un</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F  How P R1 → fir	assed st HW 25 assed st HW	<u>Un</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F  How P R1 → fir	assed st HW 25 assed st HW	<u>Un:</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect	nts: re (S2) (#HW) ts: re (S1)	Precision SP	How P R2 → fir F  How P R1 → fir	assed st HW 25 assed st HW	<u>Un:</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect Error	nts: re (S2) (#HW) ts: re (S1) ted:	Precision SP Precision	How P R2 → fix F  How P R1 → fix	assed est HW assed est HW Fixup	<u>Un:</u> HV
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect Error	nts: re (S2) (#HW) ts: re (S1) ted:	Precision SP Precision	How P R2 → fir F  How P R1 → fir	assed est HW assed est HW Fixup	<u>Uni</u> HW
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect Error  Error  Regi	nts: re (S2) (#HW) ts: re (S1) ted:	Precision SP Precision	How P R2 → fix F  How P R1 → fix	assed est HW assed est HW Fixup	Uni HW
Input Argumen  Type Structur Integer  Output Result  Type Structur  Errors Detect Error	nts: re (S2) (#HW) ts: re (S1) ted: #	Precision SP Precision	How P R2 → fix F  How P R1 → fix	assed st HW ssed st HW Fixup	Uni HW

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

	QSHAPQ
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION
	Source Member Name: QSHAPQ Size of Code Area: 74 Hw
	Stack Requirement: 18 Hw Data CSECT Size: 0 Hw
	Intrinsic X Procedure
	Other Library Modules Referenced: ETOH(DTOH), ROUND(DTOI), ITOE, ITOD
<del></del>	ENTRY POINT DESCRIPTIONS
	Primary Entry Name: QSHAPQ
	Function: Shapes data of a given type and precision to data of an explicit type and precision.
	Invoked by:  X Compiler emitted code for HAL/S construct of the form:
	Used by the INTEGER, SCALAR, MATRIX, and VECTOR shaping functions.
	Other Library Modules:
	Execution Time (microseconds): $42.6 + 31.8n$ , $n = \#$ times transferred.
	Input Arguments:  Type Precision How Passed Units
	Integer/scalar SP/DP R2 → 1st HW Integer(flag) DP R6: upper half for input data,
	Integer(count) SP lower half for output. Output Results:
	Type Precision How Passed Units
	Integer/Scalar SP/DP $R1 \rightarrow 1$ St $HW$
	Errors Detected:
	Error # Cause Fixup
	Scalar too large for integer conver- Set to maximum sion. representable
	value.  Comments: QSHAPQ is called only if more than one item of the same data type must be shaped. If only one item must undergo conversion, the conversion functions (DTOI, ETOI, ITOD, HTOE, etc.) are used. (Cont'd. on next Algorithm: The file of the same data type must be shaped.
	Algorithm: The flags for the input and output data are examined. The appropriate 'LOAD' routine is executed to load one item to be shaped. The appropriate 'STORE', or in some cases, 'CONVERT AND STORE' routine
	stores the shaped data item in the area pointed to by the destination pointer. The source pointer is updated after each load; the destination pointed is updated after each store. Data is shaped and transferred item-
	by-item. The values of the flags (R6 upper and lower) are: 0 - HW integer
	1 - FW integer
	The control of the co
	o the 1900 of the algorithm and the control of the

QSHAPQ

Comments: (Continued)

Registers Unsafe Across Call: F0,F1.

		en e	RANDOM	•
	HAL/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name	: RANDOM Siz	e of Code Area:	46	Hw
Stack Requirement:	18 Hw Dat	a CSECT Size:	2	Hw
☐ Intri	nsic	Procedure		
Other Library Modu	les Referenced;	None		
ENTRY POINT DESCRI	PTIONS			
Primary Entry Name	: RANDOM			
RANDOM		nstruct of the form		
RANDOM	Modules:			
Other Library		How Passed		<u>Un</u>
Other Library  Execution Time (minimum Arguments:	croseconds): 54.4			<u>Un</u>
Other Library  Execution Time (minimal Arguments:  Type  None  Output Results:	croseconds): 54.4  Precision -	How Passed		
Other Library  Execution Time (mi  Input Arguments:  Type  None	croseconds): 54.4			<u>Un</u>

Comments: The original SEED(F'1435') is declared as a data constant. To allow storage into this "constant" for updating SEED, the storage protection is turned off for SEED. Registers Unsafe Across Call: F0,F1,F2,F3. Algorithm: Multiply F'65539' by SEED. SEED originally = F'1435', but is updated on each pass through RANDOM. Use the least significant 32 bits of this product (SEED x 65539) to form the new SEED. If the result is  $\geq 0$ , then RESULT = new SEED. If RESULT < 0, then new SEED = RESULT-NEGMAX, where NEGMAX = X'80000000'. The positive new SEED is saved for future use, and is also converted to a floating point number for present computation of a random number. Multiply the floating point value by 2-31 to produce a random number in the range (0.0, 1.0).

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

	N		

Secondary Entry Na	me: RANDG		
Function: Generat	es random number fr zero, variance one.	om Gaussian distributi	on, mean
Invoked by:			
X Compiler emitRANDOMG		construct of the form	
Other library	modules:		
Execution Time (mi	croseconds): 575.	8	
Input Arguments: Type	Precision	How Passed	<u>Units</u>
None			
Output Results:			
Type	Precision	How Passed	<u>Units</u>
Scalar	DP P	FO/F1	
Errors Detected: Error #		Cause	Fixup
Comments: Sam	ne as RANDOM. Re	gisters Unsafe Across	Call: F0,F1,F2,F3.
		12	
		$\Sigma$ $X_i$ - 6.0, where $X_i$	is a random
number	generated by RANDOM		

		STBYTE	
HAL/S-FC LIBRAE	RY ROUTINE DESCRIPTION		
Source Member Name: STBYTE	_ Size of Code Area:	22	Hw
Stack Requirement: 0 Hw	v Data CSECT Size:	0	Hw
X Intrinsic	<pre>Procedure</pre>		
Other Library Modules Referenced:	None.		<del></del>
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: STBYTE			
Invoked by:			
Invoked by:  Compiler emitted code for HA  Compiler Execution Time (microseconds):	JSTV, CPAS, CRJSTV, CTRI	IMV byte.	
Compiler emitted code for HA  X Other Library Modules: CLA  Execution Time (microseconds):	JSTV, CPAS, CRJSTV, CTRI	IMV byte.	
Compiler emitted code for HA	JSTV, CPAS, CRJSTV, CTRI	tMV byte. byte. sed	Units
Compiler emitted code for HA  X Other Library Modules: CL  Execution Time (microseconds):  Input Arguments:  Type Precision	JSTV, CPAS, CRJSTV, CTRI 19.2 to store in upper 17.2 to store in lower  How Pas	tMV byte. byte. sed	
Compiler emitted code for HA  X Other Library Modules: CL  Execution Time (microseconds):  Input Arguments:  Type Precision Single character	JSTV, CPAS, CRJSTV, CTRI 19.2 to store in upper 17.2 to store in lower  How Pas	t byte. byte. sed  (00-upper byt x'8000' lower	

## Comments:

Registers Unsafe Across Call: Rl,R4,R5,F0,F1.

Algorithm: The flag is tested for an even or odd byte to store into. If odd (upper), the flag is set to indicate even (lower) for the probable loop that STBYTE is in. Then, the byte is inserted into the upper byte of the appropriate halfword. If the flag indicates an even byte to store into, then the byte is inserted into the lower byte of the appropriate halfword. The flag is set to 0 to indicate that the next time around in the loop, the byte will be odd. The pointer is updated to the next halfword.

			•	XTOC		
	HAL/S	-FC LIBRARY	ROUTINE DESCR	IPTION		
	Source Member Name:	XTOC	Size of Code	Area:	68	Hw
	Stack Requirement:	0 Hw	Data CSECT S	ize:	0	Hw
	Intrinsic		☐ Proced	ure		
	Other Library Modules R	eferenced:	None.			
	ENTRY POINT DESCRIPTION	S				
	Primary Entry Name:	XTOC				
	Function: Converts b	it string to	o a string of 1	hexadecimal c	haracters.	
	Invoked by:  Compiler emitted of CHARACTER GHEX  Other Library Modu	BIT_STRING)	/S construct o	40 Tible 10 8 8 8	omiar	
						) HADIS
	Execution Time (microse	33.	.9 + 32.2 ·(# c .9 · (# of let (# of digits)+	ters A-F), who	ere	•
	Input Arguments: Type	Precision		How Passed		Units
	bit string Integer(length of bit string)	SP		R6		
	Output Results: Type	Precision		How Passed		Units
	character string			R2 → descript	or	
	Errors Detected:				<b>D</b> .	
4	Error #		Cause		Fixup	

Comments: Output string length depends on input string length. The maxlength of the output area is ignored.

Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6,R7,F0,F1.

Algorithm: A character count is determined as the integer part of (bit length + 3)/4. The bit string is positioned in register pair R4-R5 with the first hexadecimal digit in bits 12-15 of R4 thus:

1) Clear R4; string right-justified in R5 on input.

2) Compute greatest multiple of 4 in 52 - bit length.

3) Use result of 2) as a shift count to shift R4-R5 left double.

Compute a halfword count for use as a loop counter:

halfword count = (1 + character count)/2

The character count is stored in the descriptor halfword as the current length of the output string. Digits are generated by shifting left 4 and stored two at a time in the output string, after converting DEU format by adding X'30' to each digit, Exit when proper number of halfwords have been stored.

been stored.

INTERMETRICS INCORPORATED · 701 CONCORD AVENUE · CAMBRIDGE. MASSACHUSETTS 02138 · (617) 661-1840

XTOC
------

Secondary Entry Name	OTOC			
Function: Converts	a bit string i	nto a string	of octal character	<b>5.</b>
Invoked by:				
X Compiler emitted	d code for HAL,	'S construct of	of the form:	
CHARACTER OCT (B				
Other library m	odules:			
Execution Time (micro	oseconds): 46.	2 + 32.3 (#	of digits), where	
	6 •	(# of digits	) = # bits + 2.	
Input Arguments:				
Type	Precision		How Passed R5	Units
bit string Integer(length of			R6	
bit string)	SP		. KO	
Output Results:				Units
Туре	Precision		How Passed	Units
character string			R2 → descriptor	ing the state of t
Errors Detected:				
Error #		Cause	Fix	<u>ıp</u>

Comments: Output string length depends on input string length. The maxlength of the output area is ignored.

Registers Unsafe Across Call: R1,R2,R3,R4,R5,R6,R7,F0,F1.

Algorithm: First, a character count is determined as the integer part of (bit length + 2)/3. The bit string is positioned in register pair R4-R5 with the first octal digit in bits 13-15 of R4 as follows: 1) Begin with R4 clear and the string right-aligned in R5. 2) Compute the shift count as 51-3(character count) and 3) shift R4-R5 left double by this amount.

Complete a halfword count for use as a loop counter as:

halfword count = (1 + character count)/2

The character count is stored in the descriptor halfword of the output string. Then, digits are generated in a loop, two at a time, by shifting R4-R5 left double 3 bits and adding X'30' to give the appropriate DEU character. As pairs of digits are assembled, they are stored into the output string, and exit is taken when the proper number of halfwords have been stored.

## 5.3.7 REMOTE Routine Descriptions

This subsection describes those routines which perform operations on REMOTE data. REMOTE data is data which may reside in a sector of AP-101 core which is neither sector 0 nor the current data sector indicated in the Program Status Word at the time the routine is called. In order to insure addressability of such data, these routines are passed, instead of pointers directly to their arguments, pointers to complete address constants, or "ZCONS", containing both the address of the argument and the number of the sector in which it resides. These complete address constants, together with a special AP-101 addressing mode, allow access to any area of AP-101 core without changing bits in the Program Status Word.

REMOTE routines are invoked (rather than the normal versions of the same routines) when at least one of the arguments of the routine has the REMOTE attribute. Since this attribute only applies to aggregate data types (VECTOR, MATRIX, STRUCUTRE and CHARACTER types), only these four types of routines have REMOTE versions.

			CASRPV	· ·
	wat /c_FC IIRDARY	ROUTINE DESCRIPTION		
				77
Source Member Na	ame: CASRPV	Size of Code Area:	86	Hw
Stack Requiremen	nt: 22 Hw	Data CSECT Size: _	2	Hw
☐ Int	rinsic	Procedure		
Other Library Mc	odules Referenced: _	None.		
ENTRY POINT DESC	CRIPTIONS		via a la companya di salah	
Primary Entry Na	casrpv			
	e character assignme	ent to temporary fro	m partitioned	input.
	mitted code for HAL/			<b>+</b>
···C <sup>2</sup> I TO J	where C2 is a R	is a temporary str	.ag.e and resur	
X Other Libra				
CPASRP				
Execution Time	(microseconds): (See	next page).		
Input Arguments	Precision	How P	assed	Unit
<u>Type</u> integer (I)	SP	R5		
integer (J)	SP	R6		4 1.74
character (C2)		$_{R4} \rightarrow z_{CON}$	→ descriptor	
character (C2) Output Results:	Precision		→ descriptor	<u>Unit</u>
character (C2)	Precision	How P		<u>Unit</u>
character (C2) Output Results: Type	Precision orary) -	How P	assed → descriptor	<u>Unit</u>
character (C2) Output Results:  Type character (tempo	Precision orary) -	$\frac{\text{How P}}{\text{R2}} \rightarrow \text{ZCON}$ $\frac{\text{Cause}}{\text{Cause}}$	assed → descriptor  Fixup	
character (C2) Output Results:  Type character (tempo	Precision orary) -	$\begin{array}{c} \text{How P} \\ \text{R2} \rightarrow \text{ZCON} \\ \\ \hline \text{Cause} \\ \text{outside current} \end{array}$	assed → descriptor	on point

Algorithm:

Set maxlength of result to 255. Test position of 1st character of partition. If < 1 then send error and set to 1.

Compare position of last character of partition. If it is > than maxlength, reset to maxlength, and send error.

(Continued on next page)

CASRPV

# Execution Time (microseconds):

If P = 0 and length(C2) = 0: 76.8

If P > 0 and I is odd:  $89.0 + 15.8 \cdot ceiling(n/2)$ 

If P > 0 and I is even:  $94.2 + 21.2 \cdot \text{ceiling}(n/2)$ 

where p = J - I + 1

n = minimum(p, 255)

## Algorithm (Con't)

Compare first and last positions. If last < first, then if input string is null do not send error. If input string is not null, send error and set result to null string. Make sure partition length does not exceed the maxlength of the destination string. If it does, truncate it. Increment character count before dividing by 2 to round resulting halfword count to next highest halfword. If position of first character of partition is odd, then transfer halfword by halfword. Otherwise, it is necessary to line characters up into right halves of halfwords by shifting.

### Comments:

Registers Unsafe Across Call: None.

REPRODUCIBILITY OF THE ORIGINAL PA

תח	R		

Secondary Entry Name	e: CASRP		
Function:	character assignm	ent to declared data, parti	tioned input.
Invoked by:			
Cl = C2 I TO J;		construct of the form: C2 are REMOTE character dat	<b>a.</b>
Other library	modules:		
Execution Time (mic	roseconds): (Se	e next below).	
Input Arguments:			
Type nteger (I) nteger (J)	Precision SP SP	How Passed R5 R6	<u>Units</u>
character(C2)		R4 → ZCON → descr	iptor -
Output Results: Type	Precision	How Passed	Units
character(Cl)		R2 → ZCON → descr	iptor -
Errors Detected: Error # 17	Same caus	<u>Cause</u> es and fix-ups as CASRPV	'ixup
Comments:			
Registers Unsa	fe Across Call:	None.	
Algorithm:			
ame as CASRPV excep not set to 255. Execution Time:	t maxlength of re	sultant string is used as p	assed and
if $p = 0$ and	length(C2) = 0:6	9.4	
if $p > 0$ and	I is odd: setup +	15.8 · (ceiling(n/2))	
if p > 0 and	I is even: setup	+ 5.2 + 21.2 · (ceiling(n/	2))
where $p = J - I +$	1		
	p > max length(Cl p > max length(Cl		
n = minimum	(p, max length(Cl		

		CASRV	
HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:CASRV	Size of Code Area:	36	Hw
Stack Requirement: 18 Hw	Data CSECT Size: _	0	Hw
☐ Intrinsic	X Procedure		
Other Library Modules Referenced:	None.		
ENTRY POINT DESCRIPTIONS			
Primary Entry Name: CASRV			
Function: Remote character assignm	ment to a temporary	ceceiver.	
	/S construct of the is a REMOTE characto		
Other Library Modules:			
Input Arguments:  Type Precision	> 0: 60.8 + 12.6 · length(C2).  How Page 1	assed	<u>Units</u>
character (C2)	$_{ m R4}$ $ ightarrow$ zcon $ ightarrow$	descriptor	
Output Results:  Type Precision character (temporary) -	$\begin{array}{c} \text{How P} \\ \text{R2} \rightarrow \text{ZCON} \end{array}$	assed → descriptor	<u>Units</u> -
Errors Detected: Error #	Cause	Fixup	
Comments: Registers Unsafe Across Cal	l: None.		
Algorithm: Sets maxlength of result to 255. maxlength of result, set current set current length of result to c to move by shifting right 1 # of If there is an odd # of character	length of result to urrent length of ing characters. Move ha	maxlength. Other out. Find # of l lfword by halfwo	erwise, nalfwords

Secondary Entry Name:	CASK		
Function: Remote ch	aracter assignmen	t to a non-temporary receiv	er.
Invoked by:			
Compiler emitted	code for HAL/S c	onstruct of the form:	
Cl = C2, where Cl	and/or C2 is a	remote character string.	
Other library mo	dules:		
Input Arguments: Type	Precision	ceiling(n/2)) + .8 (if leng maxlength(Cl)), where n = m How Passed	ninimum(length(C2), maxlength(Cl)). Units
character string		R4 → ZCON → descriptor	
Ashmut Dogulta.			
Output Results: Type	Precision	How Passed	Units
character string	- 1 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	R2 → ZCON → descriptor	
Ergors Detected:			
Error #	<u>C</u>	ause <u>Fi</u>	<u>kup</u>

Comments:

Registers Unsafe Across Call: None.

Algorithm:

Same as CASRV, but do not set maxlength of result to 255.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS FOOR

partition character by character.

**CPASR** 

Execution Time (microseconds):

77.9 + KA + KB + 
$$\sum_{k=1}^{LHP} (6.0 + KC_{LOUT+K}) + KD$$

NCHAR

+  $\sum_{k=1}^{KHP} (8.4 + KE_k + KC_{I+K-1}) + KF_{k+1}$ 

RHP

+  $\sum_{k=1}^{KHP} (6.0 + KC_{I+LIN+K-1}) + KG_{k+1}$ 

where:

LOUT = length(C1) before assignment

LIN = length(C2)

KA = 0 if J <= LOUT 13.0 if J > LOUT

LPART = J - I + 1

KB = 15.8 if LPART > 0 and LIN <= LPART
12.0 if LPART > 0 and LIN > LPART
0 if LPART = 0

LHP = I-LOUT-1 if I > LOUT + 1 0 otherwise

 $KC_X = 19.8 \text{ if } X \text{ is odd}$ 20.2 if X is even

KD = 3.2 if LHP = 0 and I is odd
4.2 if LHP = 0 and I is even
1.0 if LHP > 0 and LOUT is odd
0 if LHP > 0 and LOUT is even

NCHAR = minimum (LPART, LIN)

 $KE_X = 13.8 \text{ if } X \text{ is odd}$ 14.4 if X is even

KF = -.8 if NCHAR > 00 if NCHAR = 0

RHP = LPART-LIN if LPART > LIN
O otherwise

KG = 0 if RHP > 0 .4 if RHP = 0

Note: If any of LHP, NCHAR, RHP is zero, then that respective summation is also zero.

				•
HAI	/S-FC LIBRARY ROUT	INE DESCRIPTION		
Source Member Name:	CPASRP Size	e of Code Area:	16	Hw
Stack Requirement:	146 Hw Dat	a CSECT Size:	0	Hw
☐ Intrinsi	Lc G	Procedure		
Other Library Modules	Referenced:	CPASR, CASRPV		
ENTRY POINT DESCRIPT	IONS			
Primary Entry Name:	CPASRP			
	aracter string assi tioned output.	gnment of partit	ioned input	
Invoked by:	d code for HAL/S co	nstruct of the f	orm;	
	to 1 C1, C2 cha			
1 1 U T K 1	to 1			
orang <u>an</u> katalog Afrika da k				
Other Library Mo				
Other Library Mo	odules:			
orang <u>an</u> katalog Afrika da k	odules:	next page).		
Other Library Mo	odules: oseconds): (See	next page).		no.
Other Library Mo Execution Time (micro Input Arguments: Type	odules:	next page).  How Pas		<u>Uni</u>
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character(C2)	odules:  oseconds): (See  Precision  SP	next page).  How Pas  R4 $\rightarrow$ ZCON  R5	sed → descriptor	<u>Uni</u>
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character(C2) integer(k) integer(l)	odules: oseconds): (See  Precision SP SP	next page).  How Pas  R4 → ZCON  R5  R6		<u>Uni</u>
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character (C2)	odules:  oseconds): (See  Precision  SP	next page).  How Pas  R4 → ZCON  R5  R6 R7	→ descriptor	
Other Library Mo Execution Time (micro Input Arguments:  Type  character (C2) integer (k) integer (l) integer (l) integer (l)	odules: oseconds): (See  Precision SP SP	next page).  How Pas  R4 → ZCON  R5  R6	→ descriptor	
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character(C2) integer(k) integer(l) integer(l) integer(i) j) Output Results:	odules:  oseconds): (See  Precision  SP SP (SP   SP)	next page).  How Pas  R4 → ZCON  R5  R6 R7	→ descriptor	
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character(C2) integer(k) integer(l) integer(l) integer(l) integer(i) j) Output Results:  Type  character (C1)	odules:  oseconds): (See  Precision  SP SP SP (SP   SP)	How Pas  R4 -> ZCON  R5  R6	→ descriptor	
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character (C2) integer (k) integer (l) integer (i) integer (i) Output Results:  Type	odules:  oseconds): (See  Precision  SP SP SP (SP   SP)	next page).  How Pas R4 → ZCON R5 R6	→ descriptor	Uni
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character (C2) integer (k) integer (l) integer (l) integer (l) coutput Results:  Type  character (C1)  Errors Detected: Error #	odules: oseconds): (See  Precision	next page).  How Pas  R4 → ZCON  R5  R6	→ descriptor  sed escriptor  Fixup	Uni
Other Library Mo  Execution Time (micro  Input Arguments:  Type  character (C2) integer (k) integer (l) integer (l) integer (l) coutput Results:  Type  character (C1)  Errors Detected: Error #	odules: oseconds): (See  Precision SP SP (SP   SP) Precision	next page).  How Pas R4 → ZCON R5 R6	→ descriptor seed escriptor	Uni
Other Library Mo  Execution Time (micro  Input Arguments:	odules: oseconds): (See  Precision SP SP (SP  SP) Precision - Cau	next page).  How Pas R4 → ZCON R5 R6	→ descriptor  sed escriptor  Fixup Set out-of-bo	Uni Dunds va
Other Library Mo  Execution Time (micro  Input Arguments:	odules: oseconds): (See  Precision SP SP (SP  SP) Precision Cau cript of character bound	How Pas R4 → ZCON R5 R6	→ descriptor  sed escriptor  Fixup Set out-of-befirst or lase	Uni Dunds va Charac
Other Library Mo  Execution Time (micro  Input Arguments:	odules: oseconds): (See  Precision SP SP (SP  SP) Precision - Cau	How Pas R4 → ZCON R5 R6	→ descriptor  sed escriptor  Fixup Set out-of-befirst or lase	Uni Dunds va

branches to CASRPV. Returns, loads result address in R1, loads arg 3 and arg 7 in R5 and R6 respectively and branches to CPASR, and returns.

#### **CPASRP**

Execution Time (microseconds):

132.3 + KA + KB + KC + 
$$\sum_{k=1}^{LHP} (6.0 + \text{KD}_{OUTLEN+K})$$
 + KE  
+  $\sum_{k=1}^{NCHAR} (8.4 + \text{KF}_k + \text{KD}_{I+K-1})$  + KG  
+  $\sum_{k=1}^{RHP} (6.0 + \text{KD}_{I+INLEN+k-1})$  + KH

where:

INPART = L-K+1 if L >= K0 otherwise

INLEN = minimum(INPART, 255)

KA = 76.8 if INPART = 0 and length(C2) = 089.0 + 15.8(ceiling(INLEN(2)) if inpart > 0 and K is odd 94.2 + 21.2(ceiling(INLEN(2)) if inpart > 0 and K is even

OUTLEN = length(Cl) before assignment

KB = 0 if J <= OUTLEN 13.0 if J > OUTLEN

OUTPART = J - I + 1 if J >= I0 otherwise

> KC = 15.8 if OUTPART > 0 and INLEN = OUTPART 12.0 if OUTPART > 0 and INLEN OUTPART 0 if OUTPART = 0

LHP = I - OUTLEN - 1 if I > OUTLEN + 1 0 otherwise

 $KD_X = 9.8 \text{ if } X \text{ is odd}$ 20.2 if X is even

KE = 3.2 if LHP = 0 and I is odd 4.2 if LHP = 0 and I is even 1.0 if LHP > 0 and OUTLEN is odd 0 if LHP > 0 and OUTLEN is even

NCHAR = minimum(OUTPART, INLEN)

 $KF_X = 13.8 \text{ if } X \text{ is odd}$  14.4 if X is even

KG = -.8 if NCHAR > 00 if NCHAR = 0

(Continued on next page).



CPASRP

Execution Time (Continued):

RHP = OUTPART - INLEN if OUTPART > INLEN
O otherwise

KH = 0 if RHP > 0 .4 if RHP = 0

Note: If any of LHP, NCHAR, RHP is zero, then the respective summation is also zero.

structure(left comparand)s1 - R2 → ZCON → first Hw - structure(right comparand)s2 - R4 → ZCON → first Hw - integer(n) SP R5 - Output Results:				CSTR	
Stack Requirement: 18 Hw Data CSECT Size: 0 Hw    Intrinsic   Image: None	HAL/S-F	C LIBRARY RO	OUTINE DESCRIPTION		
Thrinsic	Source Member Name:	CSTR	size of Code Area:	18	Hw
Other Library Modules Referenced:  ENTRY POINT DESCRIPTIONS  Primary Entry Name:  CSTR  Function:  Comparison of REMOTE structures.  Invoked by:  X Compiler emitted code for HAL/S construct of the form:  IF S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds):  22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type  Precision  Structure(left comparand)S1 - R2 + ZCON + first Hw - Structure(right comparand)S2 - R4 + ZCON + first Hw - Integer (n)  SP R5 - Output Results:  Type  equal/not equal  Errors Detected:	Stack Requirement:	18 Hw I	ata CSECT Size:	0	Hw
ENTRY POINT DESCRIPTIONS  Primary Entry Name: CSTR  Function: Comparison of REMOTE structures.  Invoked by:  If S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United Structure (left comparand)S1 - R2 + ZCON + first Hw - structure (right comparand)S2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Structure (Precision R5 - R5 - Condition Code R5 - Condition C5 - C5	☐ Intrinsic		X Procedure		
Primary Entry Name:  CSTR  Function: Comparison of REMOTE structures.  Invoked by:  If Compiler emitted code for HAL/S construct of the form:  If S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United to Structure (right comparand)S1 - R2 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Precision How Passed Uni	Other Library Modules Ref	ferenced:	None		
Function: Comparison of REMOTE structures.  Invoked by:  X Compiler emitted code for HAL/S construct of the form:  IF S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United Structure (left comparand)S1 - R2 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Structure (not be supported by the structure) How Passed United Structure (not be supported by the supported by t	ENTRY POINT DESCRIPTIONS				
Invoked by:    X   Compiler emitted code for HAL/S construct of the form:   IF S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.   Other Library Modules:    Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures de compare, where n = index of first non-matching halfwords in Input Arguments:   Type	Primary Entry Name:	CSTR			
<pre>X Compiler emitted code for HAL/S construct of the form:     IF S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:     Type</pre>	Function: Compariso	n of REMOTE	structures.		
<pre> X Compiler emitted code for HAL/S construct of the form:     IF S1 = S2 Where S1, S2, or both is a REMOTE structure occupying halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure. 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:     Type</pre>					
halfwords.  Other Library Modules:  Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure, 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed Unit structure(left comparand)s1 - R2 + ZCON + first Hw - structure(right comparand)s2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed Unit equal/not equal Condition Code		de for HAL/S	construct of the f	orm:	
Execution Time (microseconds): 22.8 + 14.8n if structures compare, where n = # of halfwords in structure, 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed Universe (left comparand)s1 - R2 + ZCON + first Hw - structure(right comparand)s2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed Universe Qual/not equal Condition Code	IF S1 = S2 Whe			structure occ	upying n
n = # of halfwords in structure, 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United Structure (left comparand) S1 - R2 + ZCON + first Hw - structure (right comparand) S2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Passed United Precision Equal/not equal Condition Code	Other Library Module	as:			
n = # of halfwords in structure, 19.6 + 14.8n if structures do compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United Structure (left comparand) S1 - R2 + ZCON + first Hw - structure (right comparand) S2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Passed United Precision Equal/not equal Condition Code	and the first of the second section of the sectio				
compare, where n = index of first non-matching halfwords in Input Arguments:  Type Precision How Passed United Structure (left comparand)s1 - R2 + ZCON + first Hw - Structure (right comparand)s2 - R4 + ZCON + first Hw - integer (n) SP R5 - Output Results:  Type Precision How Passed United Passed	Execution Time (microsecon n = # of ha	onds): 22	.8 + 14.8n if struc	tures compare,	where
Type Precision How Passed United Structure (left comparand)S1 - R2 + ZCON + first Hw structure (right comparand)S2 - R4 + ZCON + first Hw integer (n) SP R5 - Output Results:  Type Precision How Passed United Structure (right comparand)S2 - R4 + ZCON + first Hw - R5 - COMPARANT CONTROL	compare,				
structure(left comparand)s1 - R2 → ZCON → first Hw - structure(right comparand)s2 - R4 → ZCON → first Hw - integer(n) SP R5 - Output Results:  Type Precision How Passed United and Precision Code  Errors Detected:		recision	How Pas	sed	Unit
structure(right comparand)s2 - R4 + ZCON + first Hw - integer(n) SP R5 - Output Results:  Type Precision How Passed United and Condition Code  Errors Detected:					
integer (n) SP R5 - Output Results:  Type Precision How Passed United and Condition Code  Errors Detected:					
Output Results:  Type Precision How Passed United the Errors Detected:  Errors Detected:		1/52 -	R4 → ZCON -	r IIIST HW	-
equal/not equal Condition Code  Errors Detected:			The state of the s	r rirst hw	
Errors Detected:	integer (n) Output Results:	SP	<b>R5</b>		
- 레크리트리아 크림 아이에 있는데 아마스 아마스 아마스 모든 아마스 모든 아마스 아마스 그는 사람이 있는 사람이 있는 사람이 아마스 가는 사람이 하는 사무를 <u>하고 있</u> 다. 아마스 아마스 아마스	integer (n) Output Results: Type P:	SP	<b>R5</b>		_ 
Error # Cause Fixup	integer (n) Output Results: Type P:	SP	R5 How Pas	ssed	_ Unit
	integer (n) Output Results: Type Properties: equal/not equal	SP	R5 How Pas	ssed Code	_ Unit
	integer (n) Output Results:  Type P: equal/not equal  Errors Detected:	SP recision	R5 <u>How Pas</u> Condition	ssed Code	<u>Uni</u>

Registers Unsafe Across Call: None.

Algorithm:

Compares structures, halfword by halfword, until two are found that are different or the end of the structure is reached. If inequality is found LM, restore R3, and set CC to 1. If equal, then LM, restore R3, and set CC to 0.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

			MSTR	
en e	AL/S-FC LIBRARY RO	NUTINE DESCRIPTION		
Source Member Name:	MSTR	Size of Code Area:	10	Hw
Stack Requirement:	<u>18</u> Hw I	Data CSECT Size:	0	Hw
☐ Intrin	sic	X procedure		
Other Library Modul	es Referenced:	None.		
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	MSTR			
Function: Moves a s	structure to or fr	om a remote locatio	n.	
Invoked by:	ed code for WAT/C	construct of the fe	~~~~	
		th is a REMOTE stru		~ ~
52 - 52, w	halfwords		coure occubirni	9 11
Other Library	Modules:			
Execution Time (mic	roseconds): 16.8	+ 15.0n		
Input Arguments:				
Type	Precision	How Pass		Uni
structure (S2) integer (n)	SP	$R4 \rightarrow ZCON \rightarrow C$	first Hw	
Titeder (U)	SF.	R5		
Output Results:				
Type structure (S1)	Precision	How Pass	<del></del>	Uni
		R2 → ZCON →	LLEST HW	•
Errors Detected:				
Error #		ause	Fixup	
Comments: Registers Uns	afe Across Call:	None.		
grandera are en la trada de la diferencia de la composición del composición de la composición de la composición del composición de la composición de la composición de la composición de la composición del composición de la composición de la composición de la composición de la composición del composición del composición de la composición del composición del composición de la composición de		en verkal in ginn ynga ynn flygyfaks flyknwyst fiff	um er samti ar gui el	
Algorithm: Moves st	ructure halfword	oy halfword.		

				MRODNE	<u> </u>
i i i i i i i i i i i i i i i i i i i	на	/S-FC LIBRARY	ROUTINE DESCRI	PTION	
	Source Member Name: N	IRODNP	Size of Code	Area: 16	Hw Hw
	Stack Requirement:	20 <b>Hw</b>	Data CSECT Si	ze: <u>0</u>	Hsw
	☐ Intrins		Y Procedure	3 · · · · · · · · · · · · · · · · · · ·	
	Other Library Module	s Referenced:	None.		
	ENTRY POINT DESCRIPT	IONS			
	Primary Entry Name:	MRODNP			
	Function: Moves a sprecision matrix.	scalar value to	all positions	of a partiti	on of a double
	Compiler emitte  MA TO B,C TO D  Other Library M  Execution Time (micr	= X; where X is REMOTE mandodules:	s a scalar, and atrix.	M is a doubl	e precision
	Input Arguments:  Type scalar integer(n) integer(m) integer(outdel)	Precision DP SP SP DP		FO R5 R6 R7	Units - - - -
	Output Results: Type	Precision	and the second s	How Passed	Units
	matrix (n,m)	DP	R2 →	$ZCON \rightarrow 0$ th el	Lement -
	Errors Detected: Error #		<u>Cause</u>		Fixu <sub>2</sub>
	Comments: Registers Unsafe	Across Call:	FO,Fl.		
	Algorithm:				

Same as MROSNP except use 4 · (# columns) as row length in halfwords, and use double precision store.

	MROSNP	
HAL/S-FC LIBRARY ROUTINE DESCR	IPTION	
Source Member Name: MROSNP Size of Code		16 Hw
Stack Requirement: 20 Hw Data CSECT S	ize:	Hw
☐ Intrinsic	re	
Other Library Modules Referenced: None.		
ENTRY POINT DESCRIPTIONS		
Primary Entry Name: MROSNP		
Function: Moves a scalar value to all positions single precision matrix.	of a partition	on of a REMOTE
Invoked by:  Compiler emitted code for HAL/S construct of the compiler emitted code for the compiler emitted code for HAL/S construct of the compiler emitted code for the code for th	and h rs a re	EMOTE single
Other Library Modules:		
Execution Time (microseconds): 22.8 + n(5.6 + 8	.6m) for an n	x m partition.
Input Arguments: Type Precision	How Passed	Units
scalar	FO DE	<u>.</u>
integer(n) SP integer(m) SP integer(outdel) DP	R5 R6 R7	
Output Results: Type Precision	How Passed	Units
	ZCON → 0 <sup>th</sup> e	lement -
Errors Detected:		Fixup
Error # Cause		
Comments:  Registers Unsafe Across Call: F0, F	. 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Algorithm: Find row length in halfwords by SL to outdel	L # columns,	1 add row length
Loop: Indexing on # rows, using BCTB		
Loop: Indexing on # columns, using BCTB Store scalar in pointed to output e	lement	
End. Add outdel (with row size) to output point	:er	

Source Member Name: MRIDNP Size of Code Area: 22  Stack Requirement: 20 Hw Data CSECT Size: 0    Intrinsic   Procedure	
Stack Requirement: 20 Hw Data CSECT Size: 0    Intrinsic   M Procedure	
Other Library Modules Referenced:    None	Hw
Other Library Modules Referenced: None  ENTRY POINT DESCRIPTIONS  Primary Entry Name: MRIDNP  Function: Mcves a partition of a double precision matrix to a partit of a double precision matrix. At least one of the matrices has the fattribute.  Invoked by:    X Compiler emitted code for HAL/S construct of the form; where M1 and M2 are double precision and at least one of M1 and M2 is F    Other Library Modules:   Where M1 and M2 are double precision and at least one of M1 and M2 is F    Other Library Modules:   Execution Time (microseconds): 28.4 + n(8.2 + 15.0m) for nxm partition	Hw
ENTRY POINT DESCRIPTIONS  Primary Entry Name: MRIDNP  Function: Moves a partition of a double precision matrix to a partit of a double precision matrix. At least one of the matrices has the Fattribute.  Invoked by:    Compiler emitted code for HAL/S construct of the form;     M1 = M2 A TO B,C TO D;   where M1 and M2 are double precision and at least one of M1 and M2 is Fattribute.   Other Library Modules:	
Function: Mcves a partition of a double precision matrix to a partition of a double precision matrix. At least one of the matrices has the fattribute.  Invoked by:    Compiler emitted code for HAL/S construct of the form:    M1 = M2	·
Function: Mcves a partition of a double precision matrix to a partit of a double precision matrix. At least one of the matrices has the fattribute. Invoked by:  [V] Compiler emitted code for HAL/S construct of the form:  M1 = M2 A TO B,C TO D;  M1 A TO B,C TO D = M2;  Other Library Modules:  Execution Time (microseconds): 28.4 + n(8.2 + 15.0m) for nxm partition  Thou Arguments:  Type Precision BC R6 R7  Type Precision BC R6  matrix(n,m) DP R2 + ZCON + Oth element  Trype Precision BC R6  matrix(n,m) DP R2 + ZCON + Oth element  Errors Detected:  Error # Cause Fixup	
of a double precision matrix. At least one of the matrices has the Fattribute.  Invoked by:	
Execution Time (microseconds): 28.4 + n(8.2 + 15.0m) for nxm partition  Input Arguments:  Type     matrix(n,m)	REMOTE matric
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	on.
Output Results:  Type matrix(n,m) $DP$ $R2 \rightarrow ZCON \rightarrow 0^{th}$ element  Errors Detected: Error #  Cause  Fixup	<u>Un</u>
matrix(n,m) DP R2 $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> element  Errors Detected: Error # Cause Fixup	Un.
Error # Cause Fixup	<u></u>
Comments:	
Registers Unsafe Across Call: F0,F1.	

Same as MR1SNP, except use double precision loads and stores and use  $4 \cdot (\# \text{ columns})$  as row length.

			MR1SNP
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION	
	Source Member Name: MRISNP	Size of Code Area:	Hv/
	Stack Requirement: 22 Hw	Data CSECT Size: _	O Hw
	☐ Intrinsic	Procedure	
(*	Other Library Modules Referenced:	None.	
	ENTRY POINT DESCRIPTIONS		
	Primary Entry Name: MRISNP		
	Function: Moves a partition of a of a signle precision matrix. Ei attribute.  Invoked by:    Compiler emitted code for HAI   M1 = M2   A TO B,C TO D; where	ther or both matrices	form: precision matrices,
	Ml A TO B,C TO D Other Library Modules:		
	Execution Time (microseconds): 28	.4 + n(8.2 + 12.6m)	for nxm partition.
	Input Arguments:  Type matrix(n,m) integer(n) integer (m) integer (indel,outdel) Output Results:  Type matrix (n,m)  Precision  Precision  SP	R4 → ZCON R R R R How P	6
	Errors Detected: Error #	Cause	Fixup
	Comments: Registers Unsafe Across Call: Algorithm: Separate indel and in halfwords, of result matrix by Add rowsize to inded Add rowsize to out Loop: Indexing on # of rows of out	e F0,F1. ORIO  d outdel into separat  y shifting left 1, #  del  tdel	
	Loop: Indexing on # of column load (single prec	ns of input, using BC ision) pointed to inpision) pointed to out	TB out element
	Add indel (with row size ad Add outdel (with row size a End.	ded) to input pointer dded) to output point	

	<del></del>			·	
			MRL	PNP	
HAL/	S-FC LIBRARY	ROUTINE DESCRIP	TION		
Source Member Name: M	RLTNP	Size of Code A	cea:	24	Hw
Stack Requirement:	22 <b>Hw</b>	Data CSECT Size	e:	0	Hw .
☐ Intrinsic		Procedure			
Other Library Modules	Referenced:	None.			
ENTRY POINT DESCRIPTIO	NS_				
Primary Entry Name:	MRlTNP				
Function: Moves a part of a single precision attribute. Invoked by:  X Compiler emitted M1 = M2 A TO B,C  M1 A TO B,C TO D Other Library Mod	matrix. At  code for HAL,  TO D; when = M2;	least one of the	the form; precisi	s has the	REMOTE
Execution Time (micros	econds): 31.2	2 + n(7.6 + 13.8m	) for nx	m partiti	on.
Input Arguments:					
Type matrix(n,m) integer(rows) integer(columns) integer(indel,outdel)	Precision  DP SP SP (SP   SP)		Passed N → Oth R5 R6 R7	element	Units - - -
Output Results:					
<pre>Type matrix (n,m)</pre>	Precision SP	$\begin{array}{c} \text{How} \\ \text{R2} \rightarrow \text{ZCC} \end{array}$	Passed N → 0 <sup>th</sup>	element	<u>Units</u>
Errors Detected:					
Error #		Cause		Fixup	

### Comments:

Registers Unsafe Across Call: FO,F1.

## Algorithm:

Same as MRISNP, except use double precision load for index alignment, and use  $4 \cdot (\# \text{ columns})$  as the length in halfwords of double precision partition.

	MRlWNP
<b>3</b>	HAL/S-FC LIBRARY ROUTINE DESCRIPTION
	Source Member Name: MRIWNP Size of Code Area: 24 Hw
	Stack Requirement: 22 Hw Data CSECT Size: 0 Hw
	☐ Intrinsic ☑ Procedure
	Other Library Modules Referenced: None.
	ENTRY POINT DESCRIPTIONS
	Primary Entry Name: MRIWNP
	Function: Moves a partition of a single precision matrix to a partition of a double precision matrix. Either or both matrices have REMOTE attribute.
	Invoked by:    X   Compiler emitted code for HAL/S construct of the form:   M1 = M2
	Input Arguments:
	Output Results: Type Precision How Passed Units matrix DP R2 $\rightarrow$ ZCON $\rightarrow$ Oth element
	Errors Detected: Error # Cause Fixup
	Comments:
	Registers Unsafe Across Call: F0,F1.
	Algorithm:
	Same as for MRISNP, except use double precision stores after zeroing

1

		· · · · · · · · · · · · · · · · · · ·	VRODN	
на	L/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	VRODN	Size of Code Area:	6	Hw
Stack Requirement: _	18 Hw	Data CSECT Size:	0	Hw .
☐ Intrins		v Procedure		
Other Library Module	es Referenced:	None.		
 ENTRY POINT DESCRIPT	FIONS			
Primary Entry Name:	VRODN			
Function: Move vector with the REM		all elements of a doub	ole precision	
		L/S construct of the f		
and the second s				
Other Library		6.4 + 9.2n, n = s	size of vector	
	roseconds): 1			
Execution Time (mic		6.4 + 9.2n, $n = s$ $\frac{\text{How Pas}}{F0}$		
Execution Time (mic Input Arguments: Type	roseconds): 1	How Pas		Units -
Execution Time (mic Input Arguments: Type scalar	Precision DP SP	How Pas F0 R5	ssed	Unit:
Execution Time (mice Input Arguments: Type scalar integer(n)	roseconds): 1	How Pas F0 R5 How Pas	ssed	<u>Unit</u>
Execution Time (mice Input Arguments: Type scalar integer(n) Output Results: Type	Precision  Precision  DP  SP  Precision	How Pas F0 R5 How Pas	ssed	<u>Unit</u>
Input Arguments:  Type scalar integer(n)  Output Results: Type vector(n)  Errors Detected: Error #	Precision  Precision  DP  SP  Precision	How Pas F0 R5 How Pas R2 → ZCON -	ssed ssed oth element	<u>Unit</u>
Input Arguments:  Type scalar integer(n)  Output Results: Type vector(n)  Errors Detected: Error #  Comments:	Precision DP SP  Precision DP DP DP DP DP	How Pas F0 R5  How Pas R2 → ZCON -	ssed ssed oth element	<u>Unit</u>
Input Arguments:  Type scalar integer(n)  Output Results: Type vector(n)  Errors Detected: Error #  Comments: Regist	Precision DP SP  Precision DP DP DP DP DP	How Pas F0 R5 How Pas R2 → ZCON -	ssed ssed oth element	
Input Arguments:  Type scalar integer(n)  Output Results: Type vector(n)  Errors Detected: Error #  Comments:	Precision DP SP  Precision DP Arecision DP	How Pas F0 R5  How Pas R2 → ZCON -	ssed ssed oth element	<u>Uni</u> - -

	VRODN	P
	HAL/S-FC LIBRARY ROUTINE DESCRIPTION	
	Source Member Name: VRODNP Size of Code Area: 10	Hw
	Stack Requirement: 18 Hw Data CSECT Size:	O Hw
	☐ Intrinsic	
	Other Library Modules Referenced: None.	
APPLICATION OF THE PERSON OF T	ENTRY POINT DESCRIPTIONS	
	Primary Entry Name: VRODNP	
	Function: Moves a scalar to all elements of a column of a do precision matrix.	uble
	<pre>Invoked by:</pre>	íx.
	Other Library Modules:	
•	Execution Time (microseconds): $21.2 + 10.0n$ , $n = length of$	vector result.
	Input Arguments:	
	Type Precision How Passed	<u>Uni t</u>
	scalar DP FO	-
	integer (n) SP R5 integer(outdel) SP R7	
	Integer(outdel) SP R7 Output Results:	
	Type Precision How Passed	Unit
	vector(n) DP R2 $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> elements of the lements of the	ment -
	Errors Detected:	
	Error #	Fixup
	Comments:	
	Registers Unsafe Across Call: F0,F1.	
	Algorithm: Same as VROSNP, except use double precision sto	res.

REPRODUCIPILITY OF THE ORIGINAL PAGE IS

			VROS	in	
	HAL/S-FC LIBRAR	Y ROUTINE DES	CRIPTION		
	Source Member Name: VROSN	Size of Co	de Area: 6		_ Hw
	Stack Requirement: 18 Hw	Data CSECT	Size:	0	_ Hw
	☐ Intrinsic	2 Proc	edure		
	Other Library Modules Referenced:	No	ne.		
	ENTRY POINT DESCRIPTIONS				
	Primary Entry Name: VROSN				
	Function: Moves a scalar to all e the REMOTE attribute.	lements of a	single precis	ion vector	with
•	Invoked by:  X Compiler emitted code for HAI				
	$\overline{V} = X$ ; X is a scalar, V a R	EMOTE single	precision vec	tor.	
	Other Library Modules:				
•	Execution Time (microseconds): 1	6.4 + n · 8.0	, n = size of	vector.	
	Input Arguments:			•	
	Type Precision scalar SP		How Passed FO		Units
	integer(n) SP		R5		
	Output Results:				
	Type Precision vector(n) SP		How Passed		Units
	Errors Detected:	R2	→ zcon → oth	'element	
	Error #	Cause		Fixup	
	Comments: Pagistana Wassa				
	Comments: Registers Unsafe Across	call: FO,F	<b>L.</b>		
	Algorithm:				
	Store elements in reverse order u and to control the loop.	sing the inpu	t length both	n as an inde	эx

ď.

				VROSI	NP	
	HAL/S	5-FC LIBRARY	ROUTINE DESC	RIPTION		
Sou	rce Member Name:	ROSNP	Size of Code	e Area:	10	Hw
Sta	ack Requirement:	18 Hw	Data CSECT	Size:	0	hw.
	<pre>Intrinsic</pre>		Y Proce	dure		
Oth	er Library Modules F	Referenced: _	none.			
ENT	RY POINT DESCRIPTION	1S				
Pri	mary Entry Name:	VROSNP				•
Fun pr	action: Moves a scal	lar to all el	ements of a	column of a s	single	
200	coked by:  Compiler emitted of  M, T = X; X a s	A			trix.	
Exe	Other Library Modu		.2 + 8.8n,	n = length	of vector	result.
Inp	out Arguments:					
	<u>Type</u> scalar integer (n)	Precision SP SP		How Passed F0 R5		Units - -
Out	integer(outdel) put Results:	SP		R7		
		Precision SP	R2 →	How Passed ZCON → 0th	element	<u>Units</u>
Err	ors Detected:					
	Error #		Cause		Fixup	
Com	ments: Registers Uns	safe Across C	all: FO,Fl.			
Alg	orithm:					
Sto	ore elements one at	a time, addir	ng outdel to	the pointer	after each	store.

Stack Requirement:  18 Hw Data CSECT Size:  0 Hw    Intrinsic   Procedure  Other Library Modules Referenced:   None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name:   VRIDN    Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:   X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.	Source Member Name: VRIDN Size of Code Area: 8 H  Stack Requirement: 18 Hw Data CSECT Size: 0 H  Intrinsic Procedure  Other Library Modules Referenced: None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by: X compiler emitted code for HAL/S construct of the form: V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.  Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments: Precision R4 + ZCON + Oth element R5  Output Results: Precision DP R1 + ZCON + Oth element  Errors Detected:				VRIDN	
Stack Requirement:  18	Stack Requirement:  18	HA	L/S-FC LIBRARY RO	OUTINE DESCRIPTION	ис	
Stack Requirement: 18 Hw Data CSECT Size: 0 Ho Data CSECT Size: 0	Stack Requirement: 18 Hw Data CSECT Size: 0 How Passed Precision Vector (n) integer (n) DP Rescription SP Vector (n) DP Rescription SP Precision Vector (n) DP Rescription None.    A Data CSECT Size: 0 How Passed Procedure   None.	Source Member Name:	VR1DN S	Size of Code Are	a: <u>8</u>	Hw
Other Library Modules Referenced: None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by: Compiler emitted code for HAL/S construct of the form: V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.  Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:  Type Precision R4 + ZCON + Oth element R5  Output Results: Type Precision How Passed R5  Output Results: Type Precision R5  Precision R5  Output Results: Type Precision R5  Execution DP R1 + ZCON + Oth element R5  Execution DP R1 + ZCON + Oth element R5  Execution DP R1 + ZCON + Oth element	Other Library Modules Referenced:  None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:  X Compiler emitted code for HAL/S construct of the form:  V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.  Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:  Type Precision How Passed R4 + ZCON + Oth element R5  Output Results:  Type Vector(n) DP R1 + ZCON + Oth element  Errors Detected:  Error # Cause Fixup  Comments:		70	Data CSECT Size:	0	Hw
Other Library Modules Referenced:  None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name:  VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:  X Compiler emitted code for HAL/S construct of the form:  V2 = V1; where V1 or V2 has been declared a REMOTE vector,  and V1, V2 are both double precision.  Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:  Type  Vector(n)  DP  R4 + ZCON + Oth element  R5  Output Results:  Type Vector(n)  DP  R1 + ZCON + Oth element  River  Errors Detected:	Other Library Modules Referenced: None.  ENTRY POINT DESCRIPTIONS  Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:  [X] Compiler emitted code for HAL/S construct of the form:  V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.  [Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:  Type Precision How Passed Precision R4 + ZCON + Oth element R5  Output Results:  Type Precision How Passed R6  Vector(n) DP R1 + ZCON + Oth element  Errors Detected:  Error # Cause Fixup  Comments:					
Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:  [X] Compiler emitted code for HAL/S construct of the form: V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.  [Other Library Modules:  Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:  Type Precision Pre	_				
Primary Entry Name: VRIDN  Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector, and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Other Library Module	es Referenced:	None.		
Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:    X   Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Function: Moves a double precision vector to a double precision vector, at least one of the vectors has the REMOTE attribute.  Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	ENTRY POINT DESCRIPT	rions			
Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Invoked by:    X   Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.   Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.    Input Arguments:   Type	Primary Entry Name:	VR1DN			
Invoked by:    X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.  Input Arguments:   Type	Invoked by:    X   Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.   Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.    Input Arguments:   Type	Function: Moves a d	double precision v	vector to a doubl	e precision vect	or, w
X Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.   Other Library Modules:    Execution Time (microseconds): 16.4 + n · 15.0, n = length of vector.    Input Arguments:   Type	Comments:    X   Compiler emitted code for HAL/S construct of the form:   V2 = V1; where V1 or V2 has been declared a REMOTE vector,   and V1, V2 are both double precision.    Other Library Modules:	at least one of the	vectors has the	REMOTE attribute	e. Agamai agamatan	
Input Arguments:  Type Precision How Passed  vector(n) DP R4 + ZCON + 0th element  integer(n) SP R5  Output Results:  Type Precision How Passed  vector(n) DP R1 + ZCON + 0th element  Errors Detected:	Input Arguments:  Type  Vector(n)  integer(n)  Output Results:  Type  Vector(n)  Vector(n)  Type  Vector(n)  DP  Precision  Precisi	and	V1, V2 are both	double precision	V CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	
Type Precision How Passed  vector(n) DP R4 + ZCON + 0th element  integer(n) SP R5  Output Results:  Type Precision How Passed  R5  Precision R5  How Passed  R5  Precision R5  Figure	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	and Other Library	V1, V2 are both o	double precision		r.
vector(n) integer(n)  Output Results:  Type vector(n)  DP R4, → ZCON → 0 <sup>th</sup> element R5  How Passed R1 → ZCON → 0 <sup>th</sup> element R1 → ZCON → 0 <sup>th</sup> element  Errors Detected:	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	and Other Library	V1, V2 are both o	double precision		or.
Output Results:  Type Precision How Passed vector(n) DP Rl + ZCON + 0 <sup>th</sup> element  Errors Detected:	Output Results:  Type Precision How Passed vector(n) DP Rl + ZCON + 0 <sup>th</sup> element  Errors Detected: Error # Cause Fixup  Comments:	and Other Library  Execution Time (mic	V1, V2 are both on Modules: roseconds): 16.4	double precision + n · 15.0, n	= length of vecto Passed	o <b>r.</b>
Type Precision How Passed vector (n) DP Rl $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> element Errors Detected:		and Other Library  Execution Time (mic Input Arguments:  Type vector(n)	V1, V2 are both of Modules: roseconds): 16.4  Precision DP	double precision $+ n \cdot 15.0, n \cdot \frac{\text{How}}{\text{R4}} \rightarrow \text{ZCON}$	= length of vecto Passed → 0 <sup>th</sup> element	
vector(n) DP Rl $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> element Errors Detected:	vector(n) DP Rl $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> element  Errors Detected: Error # Cause Fixup  Comments:	and Other Library  Execution Time (mic Input Arguments:  Type vector(n)	V1, V2 are both of Modules: roseconds): 16.4  Precision DP	double precision $+ n \cdot 15.0, n \cdot \frac{\text{How}}{\text{R4}} \rightarrow \text{ZCON}$	= length of vecto Passed → 0 <sup>th</sup> element	
Trá ses y	Error # Cause Fixup  Comments:	and Other Library  Execution Time (mic  Input Arguments:  Type vector(n) integer(n)  Output Results:	V1, V2 are both of Modules:  roseconds): 16.4  Precision DP SP	double precision $+ n \cdot 15.0$ , $n \cdot \frac{\text{How}}{\text{R4}} \rightarrow \text{ZCON}$	= length of vector  Passed  + Oth element  R5	
Error # Cause Fixup	Comments:	and Other Library  Execution Time (mic  Input Arguments:  Type vector(n) integer(n)  Output Results: Type	V1, V2 are both of Modules:  roseconds): 16.4  Precision DP SP  Precision	double precision $+ n \cdot 15.0$ , $n \cdot 15.0$ , $R4 \rightarrow ZCON$	= length of vector  Passed  + Oth element  R5  Passed	
		and Other Library  Execution Time (mic Input Arguments:  Type vector(n) integer(n)  Output Results: Type vector(n)	V1, V2 are both of Modules:  roseconds): 16.4  Precision DP SP  Precision	double precision $+ n \cdot 15.0$ , $n \cdot 15.0$ , $R4 \rightarrow ZCON$	Passed  Oth element  Passed  Oth element  Passed  Oth element	
		Other Library  Execution Time (mic  Input Arguments:  Type vector(n) integer(n)  Output Results:  Type vector(n)  Errors Detected:	V1, V2 are both of Modules:  roseconds): 16.4  Precision DP SP  Precision	double precision $+ n \cdot 15.0$ , $n \cdot 15.0$ , $R4 \rightarrow ZCON$	Passed  Oth element  Passed  Oth element  Passed  Oth element	
		Other Library  Execution Time (mic  Input Arguments:	V1, V2 are both of Modules:  roseconds): 16.4  Precision DP SP  Precision	double precision $+ n \cdot 15.0$ , $n \cdot 15.0$ , $R4 \rightarrow ZCON$	Passed  Oth element  Passed  Oth element  Passed  Oth element	

Same as VRISN, except use double precision loads and stores.

		VRL	DNP	
HAT.	/S-FC TJBRARY ROL	JTINE DESCRIPTION		
Source Member Name: V		ize of Code Area:	20	Hw
Stack Requirement:		ata CSECT Size:	0	Hw
Intrinsi		Procedure		
Other Library Modules	Referenced:	None.		
ENTRY POINT DESCRIPTI	ONS			
Primary Entry Name: _	VRLDNP			
		construct of the form		
M* J Other Library Mo  Execution Time (micro contiguous  Input Arguments:  Type vector (n)	matrix, and odules:  oseconds): 17.0n  . 17.0n + 30.4 i where n = lend  Precision  DP SP	How Passed $R4 \rightarrow ZCON \rightarrow C$ $R5$	it nor outpu out is conti	t is guous
My = V;  Other Library Mo  Execution Time (micro contiguous  Input Arguments:  Type vector (n) integer (n) integer (indel) integer (outdel)	matrix, and odules:  oseconds): 17.0n  17.0n + 30.4 in the second where n = lend of the second of th	<pre>v or M is REMOTE. + 29.6 if neither input f either input or output gth of vector.  How Passed R4 + ZCON + C</pre>	it nor outpu out is conti	t is guous <u>Un</u>
Execution Time (micro contiguous  Input Arguments:  Type vector (n) integer (n) integer (indel)	matrix, and odules:  oseconds): 17.0n  . 17.0n + 30.4 if where n = lend  Precision  DP  SP  SP  SP	+ 29.6 if neither input f either input or output of vector.  How Passed R4 + ZCON + CR5 R6 R7	ut nor outpu out is conti	t is
My J Other Library Mo Execution Time (micro contiguous Input Arguments:  Type vector (n) integer (indel) integer (outdel) Output Results:	matrix, and odules:  oseconds): 17.0n . 17.0n + 30.4 : where n = lend  Precision  DP SP SP SP SP SP	V or M is REMOTE.  + 29.6 if neither input f either input or output of vector.  How Passed R4 → ZCON → C R5 R6 R7	ut nor outpu out is conti	t is guous <u>Un</u>
M = V;  Other Library Mo  contiguous  Input Arguments:  Type vector (n) integer (n) integer (indel) integer (outdel) Output Results:  Type	matrix, and odules:  oseconds): 17.0n  17.0n + 30.4 : where n = lend  Precision  DP SP SP SP SP SP Precision  DP	+ 29.6 if neither input f either input or output of vector.  How Passed R4 + ZCON + CR5 R6 R7	ut nor outpu out is conti	t is guous <u>Un</u>
My J Other Library Mc  Execution Time (micro contiguous  Input Arguments:  Type vector (n) integer (indel) integer (outdel) Output Results:  Type vector (n)  Errors Detected: Error #	matrix, and odules:  oseconds): 17.0n  17.0n + 30.4 : where n = lend  Precision  DP SP SP SP SP SP Precision  DP	+ 29.6 if neither input if either input or output of vector.  How Passed  R4 + ZCON + C  R5  R6  R7  How Passed  R2 + ZCON + Ot	at nor outpu out is conti th element	t is guous <u>Un</u>

	HAI					
So	ource Member Name:	VR1SN	Size of Co	ode Area: 8		Hw
St	ack Requirement:	18 Hw	Data CSEC	r Size: 0		Hw .
	☐ Intrins	ic	2 Proc	edure		
Ot	ther Library Module	s Referenced:	Nor	ie.		<del></del>
EN	TRY POINT DESCRIPT	IONS				
Pr	rimary Entry Name:	VR1SN				
	unction: Moves where at least one	a single preci of the vectors	sion vector has the REM	to a single preci	ision vec	tor,
	voked by:  Compiler emitted				·••	
	VI = V2; wne			emote and V1 and V	v2 are	
	Other Library M	single precis		emote and VI and V	v2 are	
E>		single precis	ion.			
	Other Library M	single precis	ion.	, n = length of v		
	Other Library Moteration Time (microput Arguments:  Type	single precis	ion.	, n = length of v	ector.	<u>U</u> ni
	Other Library Moxecution Time (micronput Arguments:	single precis odules: oseconds):	ion.	, n = length of v	ector.	<u>Uni</u>
Ir	Other Library Moteration Time (microsput Arguments:  Type vector(n) integer(n) utput Results:	single precis odules: oseconds): <pre>Precision SP SP SP</pre>	ion.	, n = length of v  How Passed  R4 $\rightarrow$ ZCON $\rightarrow$ O <sup>th</sup> $\leftarrow$ R5	ector.	<u>Uni</u>
Ir	Other Library Mother Library Li	single precis odules: oseconds): <pre>Precision SP</pre>	ion.	, n = length of v  How Passed  R4 $\rightarrow$ ZCON $\rightarrow$ 0 <sup>th</sup> e	vector.	
Ir Ot	Other Library Moteration Time (microspet Arguments:  Type vector(n) integer(n) utput Results:  Type	single precis odules: oseconds):  Precision SP SP Precision	ion.	, n = length of v How Passed  R4 + ZCON + 0 <sup>th</sup> $\epsilon$ R5  How Passed  R2 + ZCON + 0 <sup>th</sup> $\epsilon$	vector.	
Ir Ot	Other Library Mother Library Libr	single precis odules: oseconds):  Precision SP SP Precision	ion. L6.4 + 12.6n	, n = length of v How Passed  R4 + ZCON + 0 <sup>th</sup> $\epsilon$ R5  How Passed  R2 + ZCON + 0 <sup>th</sup> $\epsilon$	vector. element	
Ir Ot En	Other Library Mother Library Libr	single precis odules: oseconds):  Precision SP SP Precision	ion. L6.4 + 12.6n	, n = length of v How Passed  R4 + ZCON + 0 <sup>th</sup> $\epsilon$ R5  How Passed  R2 + ZCON + 0 <sup>th</sup> $\epsilon$	vector. element	

REPRODUCIBILITY OF THE ORIGINAL PAGE IS TOOR

then store, each element in turn.

			VRISNP	
	HAL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name	VRISNP	Size of Code Area:	20	Hw
Stack Requirement:	18 Hw	Data CSECT Size: _	0	Hw .
☐ Intri	insic	y Procedure		
Other Library Modu	les Referenced:	None.		
ENTRY POINT DESCRI	PTIONS			
Primary Entry Name	VRISNP			
elements of source REMOTE attribute. Invoked by:	e or receiver are	on vector to a single on not contiguous and	at least one ha	cor when as the
V = M.,J;  M.,J = V;  Other Library		Lared single precision matrix, and Vec.		
contiguo W		<pre>6n + 30.4 if either 5 if neither input nor 6f vector.</pre>		
Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel) Output Results:	Precision SP SP SP SP SP	How Pa	ssed oth element	Units - - -
Type	Precision	How Pa		Units
vector	''	R2 → ZCON →	0 <sup>th</sup> element	-
Errors Detected: Error #		Cause	Fixup	
Comments:				
그는 이렇게 있는데 그렇게 아들을 걸어 보는데	rs Unsafe Across	call: FU,FL.		
Algorithm:				
<pre>If outdel = 0, If indel = 0,</pre>				
Loops 'length' ti pointer and outde	mes, moving one e l to output point	lement each loop. A er after each move.	dds indel to in	put

5-358

			VRLTN	
	HAL/S-FC LIBRARY	ROUTINE DESCRI	IPTION	
Source Member Name:	VRITN	Size of Code	Area: 8	Hw
Stack Requirement:	18 Hw	Data CSECT Si	.ze:0	Hw
☐ Intrin	nsic	2 Procedu	re	
Other Library Modul	es Referenced:	None	•	
ENTRY POINT DESCRIP	TIONS			
Primary Entry Name:	VRlTN			
Function: Massa	double precision	n vector to a s	ingle precision vo	ctor,
<pre>Invoked by:</pre>	of the vectors	has the REMOTE	attribute.	
Invoked by:    X Compiler emitt   V1 = V2; wher   vec	ed code for HAL, e Vl is a single tor, and at leas	has the REMOTE  /S construct of  e precision vect  st one of VI and	the form: tor, V2 is a doubled V2 is REMOTE.	e precisio
Invoked by:    X Compiler emitt   V1 = V2; where   Vec	ed code for HAL, e Vl is a single tor, and at leas	has the REMOTE  /S construct of  e precision vect  st one of VI and	the form: tor, V2 is a doubled V2 is REMOTE.	e precisio
Invoked by:  X Compiler emitt V1 = V2; wher vec Other Library  Execution Time (mic	ed code for HAL, e Vl is a single tor, and at leas Modules: roseconds): 16.	has the REMOTE  /S construct of e precision vect st one of V1 and  4 + 13.8n,	<pre>the form: tor, V2 is a double d V2 is REMOTE.  n = length of vectors</pre>	e precisio
where at least one  Invoked by:  X Compiler emitt  V1 = V2; wher  vec  Other Library  Execution Time (mic	ed code for HAL, e Vl is a single tor, and at leas	has the REMOTE  /S construct of precision vect st one of V1 and  4 + 13.8n,	the form: tor, V2 is a doubled V2 is REMOTE.	e precisio
<pre>Invoked by:     X Compiler emitt     V1 = V2; wher     vec     Other Library  Execution Time (mic Input Arguments:</pre>	ed code for HAL, e Vl is a single tor, and at leas Modules:  roseconds): 16.  Precision DP	has the REMOTE  /S construct of precision vect st one of V1 and  4 + 13.8n,	the form; tor, V2 is a double V2 is REMOTE.  n = length of vector ow Passed CCON \rightarrow 0th element	e precisio
where at least one  Invoked by:  X Compiler emitt  V1 = V2; wher  vec  Other Library  Execution Time (mic  Input Arguments:  Type  vector (n)  integer(n)	ed code for HAL, e Vl is a single tor, and at leas Modules:  roseconds): 16.  Precision DP	has the REMOTE /S construct of e precision vector one of V1 and $4 + 13.8n$ , $\frac{H}{R4} \rightarrow \frac{H}{2}$	the form; tor, V2 is a double V2 is REMOTE.  n = length of vector ow Passed CCON \rightarrow 0th element	e precisio
Invoked by:    X Compiler emitt   V1 = V2; wher   Vec   Other Library    Execution Time (mic   Input Arguments:   Type     vector (n)     integer(n)  Output Results:   Type	ed code for HAL, e Vl is a single tor, and at leas Modules:  roseconds): 16.  Precision DP SP  Precision	has the REMOTE /S construct of e precision vector one of V1 and $4 + 13.8n$ , $\frac{H}{R4} \rightarrow \frac{H}{2}$	the form; tor, V2 is a double V2 is REMOTE.  n = length of vector ow Passed CCON + Oth element R5	e precisio

Same as VRISN, except use double precision loads.

		V	RlTNP	•
H.	AL/S-FC LIBRARY	ROUTINE DESCRIPTION		
Source Member Name:	<u> </u>	Size of Code Area:	20	Hw
		Data CSECT Size:	0	Hw
· · · · · · · · · · · · · · · · · · ·		4. 4 <u></u>		
[] Intrin	51 <b>C</b>	X Procedure		
Other Library Modul	es Referenced:	None.		
ENTRY POINT DESCRIP	rions .			
Primary Entry Name:	VRLTNP			
of them has the REM Invoked by:    X   Compiler emitt   V = M   , J;   where   Other Library   Execution Time (mic   Input Arguments:   Type   Vector (n)	OTE attribute.  ed code for HAL  V is a single p  matrix, and V of  Modules:  roseconds): 15.8  15.8n + 29.6 if	Ar are not contiguous,  As construct of the form of th	orm; a double pre out or output out is contig	cision
integer(n) integer(indel) integer(outdel)	SP SP	R6 R7		-
Output Kesults: Type	Precision	How Pass	sed	Unit
vector	SP	$R2 \rightarrow ZCON \rightarrow C$		
Errors Detected:				
Error #		Cause	Fixup	
	• 13			
Comments:				
Commence:	Across Call:	FO,F1.		
Algorithm:				
	of if indol -	O, sets it to 4, and do	as double	
precision loads.	-Dr TT THEET -	o, sees he so at and de		

				VR1WN	-
	HAL/S-FC	LIBRARY RO	OUTINE DESCRIPTION		
	Source Member Name: VRIWN		Size of Code Area:		Hw
	Stack Requirement: 1	-		0	
	☐ Intrinsic		en e		Hw .
			Procedure		
	Other Library Modules Refer	enced:	None.	***	
	ENTRY POINT DESCRIPTIONS				
	Primary Entry Name: VRLWN				
	Function: Moves a single where at least one of the v	precision ectors has	vector to a double the REMOTE attrik	e precision ve	ctor,
	Invoked by:  Compiler emitted code  V1 = V2; V1 or V2 remot	for HAL/S e, Vl doub	construct of the model of the m	Form: V2 single pred	cision.
	Other Library Modules:  Execution Time (microseconds	s): 20.6	+ 13.8n, n = 1en	gth of vector	
	Input Arguments:				
		Lsion	$ \begin{array}{c} \text{How Pas} \\ \text{R4} \rightarrow \text{ZCON} \rightarrow 0 \\ \text{R5} \end{array} $		Units - -
•	Output Results:				
	Type Preci	sion	How Pas	sed	Units
	vector(n) DP		$R2 \rightarrow ZCON \rightarrow 0$	th element	
	Errors Detected:				
	Error #	<u>C</u> ŧ	use	Fixup	
	Comments:				
	Registers Unsafe Across Cal	1: FO,F1.			
	Algorithm:				
	Same as VRISN, except use do	uble preci	sion store with lo	ow half of flo	ating

REPRODUCIBILITY OF THE ORIGINAL PAGE IS

		VRIV	INP	
H	AL/S-FC LIBRARY ROU	TINE DESCRIPTION		
Source Member Name:	VRIWNP Si	ze of Code Area:	22	Hw
Stack Requirement:	18Hw Da	ta CSECT Size:	0	Hw
☐ Intrin	nsic	X Procedure		
Other Library Modul	es Referenced:	None.		
ENTRY POINT DESCRIP	PTIONS			
Primary Entry Name:	VRIWNP			
where V		construct of the formation vector, M is a s		sion mat
ar الرحم	nd v or M is REMUTE.			
Other Library				
Other Library				
Execution Time (mic	Modules: croseconds): 15.8n	+ 31.2 if either inp	ıt or outpu contiguous	t is con
Execution Time (mic 15.8	Modules: croseconds): 15.8n	+ 31.2 if either inportant to the contract of the contract is	contiguous	
Execution Time (mid 15.8) Input Arguments: Type	Modules:  croseconds): 15.8n  3n + 32.0 if neither  Precision	+ 31.2 if either inport in the control of the contr	contiguous	t is con • Unit
Execution Time (mic 15.8 Input Arguments: Type vector(n) integer(n)	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP	+ 31.2 if either input input nor output is  How Passee  R4 $\rightarrow \overline{\text{ZCON}} \rightarrow 0$ th  R5	contiguous	
Execution Time (mic 15.8 Input Arguments: Type vector(n) integer(n) integer(indel) integer(outdel)	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP	+ 31.2 if either input input nor output is  How Passes  R4 $\rightarrow \frac{\text{How Passes}}{\text{ZCON}} \rightarrow 0 \text{ th}$	contiguous	
Execution Time (mid 15.8)  Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel) Output Results:	Modules:  croseconds): 15.8n  3n + 32.0 if neither  Precision  SP  SP  SP  SP  SP  SP	+ 31.2 if either input input nor output is  How Passes  R4 $\rightarrow \frac{\text{How Passes}}{\text{ZCON}} \rightarrow \text{Oth}$ R5  R6  R7	contiguous  delement	
Execution Time (mic 15.8 Input Arguments: Type vector(n) integer(n) integer(indel) integer(outdel)	Modules:  croseconds): 15.8n  3n + 32.0 if neither  Precision  SP SP SP SP	+ 31.2 if either input input nor output is  How Passes  R4 $\rightarrow \frac{\text{How Passes}}{\text{ZCON}} \rightarrow \text{Oth}$ R5  R6	contiguous  delement	Unit
Execution Time (mic 15.8 Input Arguments: Type vector(n) integer(n) integer(indel) integer(outdel) Output Results: Type vector(n)	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP SP SP Precision	+ 31.2 if either input input nor output is  How Passee  R4 + ZCON + Oth R5 R6 R7  How Passee	contiguous  delement	Unit
Execution Time (mid 15.8)  Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel)  Output Results:  Type	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP SP SP Precision	+ 31.2 if either input input nor output is  How Passee  R4 + ZCON + Oth R5 R6 R7  How Passee	contiguous  delement	
Execution Time (mic 15.8 Input Arguments: <u>Type</u> vector(n) integer(n) integer(indel) integer(outdel) Output Results: <u>Type</u>	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP DP	+ 31.2 if either input input nor output is  How Passee  R4 + ZCON + Oth R5 R6 R7  How Passee	contiguous  delement	<u>Uni</u>
Execution Time (mic 15.8 Input Arguments: Type vector(n) integer(n) integer(indel) integer(outdel) Output Results: Type vector(n) Errors Detected:	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP DP	+ 31.2 if either input is input nor output is  R4 + ZCON + Oth R5 R6 R7  How Passes R4 + ZCON + Oth	element	Uni.
Execution Time (mic 15.8  Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel)  Output Results:  Type vector(n)  Errors Detected: Error #	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP DP	+ 31.2 if either input is input nor output is  R4 + ZCON + Oth R5 R6 R7  How Passes R4 + ZCON + Oth	element	Unit
Execution Time (mic 15.8  Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel)  Output Results:  Type vector(n)  Errors Detected: Error #  Comments:	Modules:  croseconds): 15.8n  3n + 32.0 if neither  Precision  SP  SP  SP  SP  SP  DP  Co	+ 31.2 if either input input nor output is  How Passee R4 + ZCON + 0th R5 R6 R7  How Passee R2 + ZCON + 0th	element	Unit
Execution Time (mic 15.8  Input Arguments:  Type vector(n) integer(n) integer(indel) integer(outdel)  Output Results:  Type vector(n)  Errors Detected: Error #  Comments:	Modules:  croseconds): 15.8n  Bn + 32.0 if neither  Precision  SP SP SP SP SP SP SP DP	+ 31.2 if either input input nor output is  How Passee R4 + ZCON + 0th R5 R6 R7  How Passee R2 + ZCON + 0th	element	Unit

#### 6.0 SYSTEM INTERFACES

This section deals with characteristics and behavior of the HAL/S-FC compiler as related to the environment in which the compiler operates. Specifically, these items are in relation to the host computer in which the compiler is executed.

## 6.1 <u>Internal System Interfaces</u>

The HAL/S-FC compiler is designed to operate under OS/360 MVT or an equivalent operating system (such as OS/VS2 on IBM 370 equipment.) The compiler was developed under Release 21.6 of OS and uses many of the features of that system.

#### 6.1.1 Macro Instructions

All operating system communication is performed via standard assembler language macro instructions as provided with OS MVT. The following list contains the names of all macros executed directly by the HAL/S-FC compiler.

ABEND	BLDL	CHECK	CLOSE	DCB
DCBD	DELETE	FIND	FREEMAIN	FREEPOOL
GET	GETBUF	GETMAIN	LOAD	NOTE
OPEN	POINT	PUT	READ	RETURN
SAVE	SPIE	STIMER	STOW	TIME
TTIMER	WRTTE			

The forms of some of these macros require further explanation:

- FREEMAIN All FREEMAIN macros use the SP parameter to indicate subpool 22. Both freeing of single elements of storage and freeing of an entire subpool are performed.
- GETMAIN All requests for main storage are made
  with the SP operand specifying subpool 22.
  GETMAINs are done for both single elements
  of storage of specific size and once, during
  compiler initialization, for a variable
  region using the VC form of GETMAIN. This
  initialization GETMAIN obtains the largest
  contiguous element of memory available in
  the job step region. This memory (assigned
  to subpool 22) is used to hold executable
  compiler code and as a data area for the compiler.

# REPRODUCIBILITY OF THE ORIGINAL PAGE IS POSSE

STIMER - The STIMER macro with the TASK option is used to start an accounting of CPU time used by the compiler.

TTIMER - The TTIMER macro is used to test the TASK interval timer as started by the STIMER macro to determine elapsed CPU time at various points in a compilation.

## 6.1.2 Dynamic Invocation of the Compiler

The HAL/S-FC compiler may be dynamically invoked by another processing program. The details of this interface are controlled by the HAL/SDL Interface Control Document.

The dynamic invocation capability allows:

- specification of a parameter string to be acted upon by the compiler,
- specification of an alternate DDNAME list for those DD cards referenced by the compiler, and
- specification of communication areas in which the compiler will supply information to the invoking program.

The compiler takes the following actions to restore its environment upon return to the program which performed the invocation.

- All DCB's opened by the compiler are closed and any automatically acquired buffers are FREEPOOLed.
- All GETMAINED storage is FREEMAINED.
- The SPIE exit (if any) is restored to its status upon entering the compiler.

## 6.1.3 OS/360 Access Methods

In performing input/output processing the HAL/S-FC compiler uses the OS Data Management Access Methods:

BSAM OSAM BPAM

No other access methods are used, and all datasets manipulated by the compiler are standard OS/360 datasets.

## 6.2 User or External System Interfaces

The majority of ways in which users of the HAL/S-FC compiler interact with the compiler are described in Sections 2 through 5 of this document. However, the primary vehicle for user communication with this system is Job Control Language which is a part of the compiler's interface to the system in which it operates. This subsection describes the two areas of external or user interfaces to the system:

- 1) user-defined options acted upon by the compiler, and
- 2) The JCL with which the user defines the compiler's data and hence the environment in which the compiler is to operate.

## 6.2.1 User-defined Options

The HAL/S-FC compiler has a number of optional features which may be exercised by the user. These options are indicated via keyword parameters passed to the compiler in the standard OS/360 method. The options are either passed to the compiler during dynamic invocation as described in the HAL/SDL ICD, or are passed via the PARM field on the EXEC card in the JCL invoking the compiler. A list of these options and their effects may be found in Appendix A.

## 6.2.2 Job Control Language Specification

JCL is the means by which any user of the compiler defines the set of data upon which the compiler is to operate. This JCL is therefore the first interface of the user and the compiler. Once this set of data is specified, all other interfaces with the user are through this data in the manner described in preceeding chapters. The remainder of this subsection consists of two parts: 1) a listing of some typical JCL for compiler invocation; and 2) a chart describing the uses, presumed attributes, and access methods for all DD cards.

```
//HALFC
            PROC OPTION=
                                                                              00010000
//HAL
            EXEC PGM=MONITOR, REGION=350K, TIME=1,
                                                                              00020000
               PARM= '&OPTION
                                                                              00030000
//STEPLIB
            DD DISP=SHR, DSN=HALS101. MONITOR
                                                                              00040000
//PROGRAM
           DD DISP=SHR, DSN=HALS101.COMPILER
                                                                              00050000
//SYSPRINT DD SYSOUT=A, DCB=(RECFM=FBA, LRECL=133, BLKSIZE=3458)
                                                                              00060000
//LISTING2 DD SYSOUT=A, DCB=(RECFM=FBA, LRECL=133, BLKSIZE=3458)
                                                                              00070000
//OUTPUT3
           DD UNIT=SYSDA, DISP=(MOD, PASS), SPACE=(CYL, (1,1)),
                                                                              00080000
11
               DCB=(RECFM=FB, LRECL=80, BLKSIZE=400),
                                                                              00090000
               DSN=&&HALOBJ
                                                                              00100000
//OUTPUT4
           DD SYSOUT=B, DCB=(RECFM=FB, LRECL=80, BLKSIZE=400)
                                                                              00110000
           DD DISP=(MOD, PASS), DSN=&&HALSDF, SPACE=(TRK, (2,2,1)),
//OUTPUT5
                                                                              00120000
               DCB=(RECFM=F, LRECL=1680, BLKSIZE=1680), UNIT=SYSDA
                                                                              00130000
           DD DISP=(MOD, PASS), DSN=&&TEMPLIB, SPACE=(TRK, (2,2,1)),
//OUTPUT6
                                                                              00140000
              DCB=(RECFM=FB, LRECL=80, BLKSIZE=1680), UNIT=SYSDA
//
                                                                              00150000
           DD DUMMY, DCB=(RECFM=FBM, LRECL=133, BLKSIZE=133)
//OUTPUT7
                                                                              00160000
           DD DISP=SHR, DSN=HALS101.ERRORLIB
//ERROR
                                                                              00170000
           DD UNIT=SYSDA, SPACE=(CYL, 3)
//FILE1
                                                                              00180000
           DD UNIT=SYSDA, SPACE=(CYL, 3)
//FILE2
                                                                              00190000
//FILE3
           DD UNIT=SYSDA, SPACE=(CYL, 3)
                                                                              00200000
//FILE5
           DD UNIT=SYSDA, SPACE=(CYL, 3)
                                                                              00210000
//FILE6
           DD UNIT=SYSDA, SPACE=(CYL, 3)
                                                                              00220000
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS YOUR

Typical JCL for Compiler Invocation

## Compiler DDNAMES, Uses, and Requirements

DDNAME	FUNCTION	DEVICE REQUIREMENTS	LRECL	RECFM	BLKSIZE	BUFNO <sup>3</sup>	DSORG	ACCESS METHOD, MACRF
PROGRAM	executable :	<ul><li>direct access</li><li>magnetic tape</li></ul>	7200	F	7200	0	PS	BSAM, R
SYSPRINT	Primary listing	· printer · intermediate storage	133	FBA	3458 <sup>1</sup>	<b>1</b>	PS	QSAM, PL
LISTING2	Secondary unformat- ted listing	<ul><li>printer</li><li>intermediate storage</li></ul>	133	FBA	3458 <sup>1</sup>	1	PS	QSAM, PL QSAM, PL
OUTPUT3	object module output	<ul><li>direct access</li><li>magnetic tape</li><li>card punch</li></ul>	80	FB	400 <sup>1</sup>	1	PS	QSAM, PL QSAM, PL QSAM, PL QSAM, PL
OUTPUT4	duplicate object module output	<ul><li>direct access</li><li>magnetic tape</li><li>card punch</li></ul>	80	FB	400 <sup>1</sup>	1	PS	QSAM, PL
OUTPUT5	Simulation data file output	· direct access	1680 <sup>4</sup>	F <sup>4</sup>	1680 <sup>4</sup>	0	PO	BPAM, W
OUTPUT6	Template search and creation	· direct access	80 <sup>2</sup>	FB	, 1680 <sup>2</sup>	1	PO	BPAM, WR
OUTPUT7	pseudo-assembly listing for link- edit ABSLIST function	· direct access · magnetic tape	133	FBM	3458 <sup>1</sup>	1	PS	BSAM, PL
ERROR	Compiler error message retrieval	· direct access	80	FB	400	1	PO	BPAM, R
FILEL	HALMAT work file	· direct access	7200	F	7200	0	PS	BSAM, RWP

## Compiler DDNAMES, Uses, and Requirements (Con't)

DDNAME	FUNCTION	DEVICE REQUIREMENTS	LRECL	RECFM	BLKSIZE	BUFNO <sup>3</sup>	DSORG	ACCESS METHOD,
FILE2	Literal communication	direct access	1560	F	1560	0	PS	ESAM, RWP
FILE3	Phase I Init/Const work area Phase II code gen. work area	· direct access	1600	F	1600	0	PS	BSAM, RWP
FILE5	Phase III paging area	direct access	1680	F	1680	0	PS	BSAM, RWP
FILE6	Statement data communication area	direct access	512	F	512	0	PS	BSAM, RWP
SYSIN	Primary source input	<ul><li>card reader</li><li>intermediate storage</li></ul>	80 <= LRECL < 132	FB	legal mul- tiple of LRECL	1	PS	QSAM, GL
INCLUDE	Secondary source input	· direct access	80 <= LRECL <= 132	FB	legal mul- tiple of LRECL <sup>1</sup>	1	PO	BPAM, R
ACCESS	ACCESS Rights control	· direct access	80 <sup>2</sup>	FB	1680 <sup>2</sup>	1	PO	BPAM, R

#### Notes:

- 1. BLKSIZE value may be altered by user to any installation-legal value.
- 2. Compiler will use LRECL and BLKSIZE supplied by user.
- 3. BUFNO may be specified by user for any PS type datasets.
- 4. Defaults are shown; Records are always written as 1680 blocks but user-supplied attributes will be retained.

#### Appendix A

## Compile-Time JCL Options

This Appendix describes the compiler options which may be coded in the PARM field of the EXEC card in the Job Control Language invoking the compiler. In all cases, options are separated in the PARM field by commas. If an option is referenced more than once in a PARM field, the last reference (scanning left to right) will be used to determine the option's setting.

There are two general classes of options recognized by the compiler: Type 1 options having a binary value of "on" or "off", and Type 2 options having a numeric or string value.

## Type 1 Options

Type 1 options are controlled by keywords in the PARM field. The appearance of the keyword indicates that the option is to be "on" during the compilation unless the keyword is preceded by the characters "NO" in which case the option is "off". Some Type 1 options have alternate, shorter spellings which may be used interchangably with the standard keywords.

When a Type 1 option has an alternate form, the negative or "off" value (equivalent to adding 'NO' to the standard keyword) is specified by preceeding the alternate form with the character 'N'. The 'NO' and 'N' notations may only be used with the standard and alternate forms respectively. For example, the LIST option has the alternate form L. If the negative is to be specified, it may be done as NOLIST or NL; NLIST or NOL will not be recognized.

The following Type 1 options are recognized. The default settings shown are used in the absence overriding PARM field specifications.

Keyword	Alternate	Default	Function
LISTING2	L2	off	Causes unformatted source listing to be generated.
DUMP	DP	off	Requests the compiler to produce a memory dump if certain internal compiler errors occur.
LIST	L	off	Produces an assembly listing from Phase II of the compiler.
TRACE	TR	on on	Causes the generation of a link to the HSS end-of-statement routine in the object module. Enables Real Time execution and debugging.
DECK	D	off	Controls production of an additional object deck on the OUTPUT4 DD card.
TABLST	TL	off	Causes Phase III of the compiler to produce formatted dump of the simulation data file (SDF).
SRN	none	off	Causes the compiler to omit the last eight columns or characters from the source scanning. These columns are then used to print infor- mation on the listing.
TABLES	TBL	on	Controls generation of Simu- lation Data Files.
ADDRS	<b>A</b>	off	Indicates the presence of statement address information in the Simulation Data Files.
ZCON	2	On	Indicates external linkage conventions to be used (via Z_CONs or direct).
TABDMP	TBD	off	Causes Phase III of the compiler to produce a hexadecimal dump of the simulation data file.

# REPRODUCIBILITY OF THE ORIGINAL PAGE

Keyword	Alternate	Default	Function
SDL	none	off	Informs the compiler that it is operating within the SDL. ACTIONS specific to SDL operations are keyed to this option such as inclusion of SRN, Change Authorization Field and Source record revision indicator on primary listing.
FCDATA	FD	off	Causes HAL/S-360 data to be allocated using a halfword as the basic memory unit. This causes data area allocation which maps directly into HAL/S-FC data allocation.
ZCON	<b>Z</b>	on	Causes calls to out-of-line routines to be performed via long indirect address constants (ZCONs).
SCAL	SC	on	Allows use of SCAL/SRET instructions for subroutine linkage. If off, BAL linkage is used. SCAL is meaningless if NOMICROCODE is specified.
MICROCODE	MC	on	Allows use of instructions which only exist on late versions of the Space Shuttle GPC. This includes SCAL, SRET, MVS, MVH and BIX. Use of SCAL and SRET may be separately controlled with the SCAL option.
SREF	SR	off	Causes special processing of user-defined symbols which appear within an EXTERNAL COMPOOL template which is included in another compilation. Any items in such a COMPOOL which are not referenced by the primary compilation unit are not printed in the symbol table listing.

## Type 2 Options

Type 2 options have "values" which may be altered by the user. The values are specified by including the pseudo-assignment statement:

## ..., <type 2 opt>=<value>,...

in the PARM field where <type 2 opt> is one of the legal type 2 options, and <value> is the value to be used during compiler execution. The form of <value> is determined by the specific options. Some Type 2 options have alternate, shorter spellings which may be used interchangably with the standard forms.

The following Type 2 options are recognized. The default values shown are used in the absence of overriding PARM field specifications.

Standard	Alternate	Default	<u>Function</u>
PAGES=	<b>p=</b>	250	Sets the maximum page num- ber to be allowed in genera- tion of the primary compila- tion listing.
LINECT=	LC=	. 59	Sets the maximum number of lines which will be printed on any one page of either the primary or secondary source listing.
TITLE=	<b>T=</b>	null	Specifies 1 to 60 characters used by the compiler when printing header information at the top of each page of the listing.
SYMBOLS=	SYM=	200	Specifies the size of the compiler's symbol table.
MACROSIZE=	MS=	500	Specifies the maximum number of characters allowed in text of macro definitions.

Standard	Alternate	Default	Function
LITSTRINGS=	LITS=	2000	Specifies the maximum total number of characters allowed in character literals in a table.
COMPUNIT=	CU=	0	Specifies a compilation unit number to identify the unit of compilation. The number is made available in the SDF and in the Block Data Areas for code blocks in a HAL/S-FC compilation.
XREFSIZE=	XS=	2000	Specifies the number of cross reference table entries allocated by the compiler. Each entry used 4 bytes of storage.
CARDTYPE=	CT=	null	Specifies pairs of characters which define a mapping of arbitrary input record types (column 1 of the record) into the standard types (E,M,S,C,D, and blank). E.g. CT=XYCM would cause any 'X' records to be compiled as comments and any 'T' records to be compiled as 'M' records.
LABELSIZE=	LBLS=	1200	Specifies the maximum number of internal label points which will be maintained by the code generator.
DSR=	none	<b>1</b>	Specifies the value to be used for the data sector register in the right hand halfword of the R2 operand of the MVH instruction. The compiler will use this value explicitly when it is not possible to use a standard Z-type address constant in which the DSR field is filled in by the linkage editor.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POSS

#### Appendix B

#### Compiler Directives

The following compiler directives have been defined for the HAL/S-FC compiler.

- a) The DEVICE directive has the form:
  - D DEVICE CHANNEL=n <option>

This option is accepted by the HAL/S-FC compiler to insure compatability of source input with the HAL/S-360 compiler. It has no effect in HAL/S-FC operation.

- b) The INCLUDE directive has the form:
  - D INCLUDE <name> <option>

or

D INCLUDE TEMPLATE <unit name > <option >

This directive names a member of an include library as defined in Section 2.2. The <option> may be "NOLIST" or null. The "NOLIST" option indicates that the included text is not to be listed.

- c) The PROGRAM directive has the form:
  - D PROGRAM ID=<id>

This directive provides a Program Identification Name to be used by the compiler to determine access rights to controlled resources as described in Section 2.3.

- d) The version directive has the form:
  - D VERSION @

This directive provides version information for templates. It is generated and checked automatically by the compiler.

REPRODUCTIONS OF THE

e) The EJECT directive has the form:

D EJECT

and causes the compiler to eject a page before resuming generation of the primary source listing. The EJECT directive record is not printed on the listing.

f) The SPACE directive has the form:

D SPACE[<n>]

where <n> is an integral number.

This directive causes the compiler to skip <n> lines prior to generation of the next line in the primary source listing. The <n> may be omitted in which case one line is skipped. If <n> is greater than 3, 3 lines are skipped. The SPACE directive is not printed on the listing.

#### Appendix C

#### Error Classifications

Note: "b" denotes a blank.

#### CLASS A: ASSIGNMENT STATEMENTS

A ARRAY ASSIGNMENT

V COMPLEX VARIABLE ASSIGNMENT b MISCELLANEOUS ASSIGNMENT

## CLASS B: COMPILER TERMINATION

B HALMAT BLOCK SIZE

N NAME SCOPE NESTING

S STACK SIZE LIMITATIONS
T TABLE SIZE LIMITATIONS

X COMPILER ERRORS

b MISCELLANEOUS

#### CLASS C: COMPARISONS

b GENERAL COMPARISONS

#### CLASS D: DECLARATION ERRORS

A ATTRIBUTE LIST

C · STORAGE CLASS ATTRIBUTE

D DIMENSION

F FUNCTION DECLARATION

I INITIALIZATION

LOCKING ATTRIBUTE

Q STRUCTURE TEMPLATE TREE ORGANIZATION S FACTORED/UNFACTORED SPECIFICATION

T TYPE SPECIFICATION



b b	UNDECLARED DATA MISCELLANEOUS
CLASS E:	EXPRESSIONS
A	ARRAYNESS
В	BIT STRING EXPRESSIONS
Č	CROSS PRODUCT
D	DOT PRODUCT LIST EXPRESSIONS
L M	MATRIX EXPRESSIONS
0	OUTER PRODUCT
v	VECTOR EXPRESSIONS
b	MISCELLANEOUS EXPRESSIONS
CLASS F:	FORMAL PARAMETERS & ARGUMENTS
D	DIMENSION AGREEMENT
N	NUMBER OF ARGUMENTS
S	SUBBIT ARGUMENTS
$oldsymbol{T}$	TYPE AGREEMENT
	<ul> <li>A transfer of the control of the contr</li></ul>
CLASS G:	STATEMENT GROUPINGS (DO GROUPS)
В	BIT TYPE CONTROL EXPRESSION
C	CONTROL EXPRESSION
E	EXIT/REPEAT STATEMENTS
$\mathbf{L}$	END LABEL
V	CONTROL VARIABLE
CLASS I:	IDENTIFIERS
CHADO 1.	
$\mathbf{L}$	LENGTH
R	REPLACED IDENTIFIERS OUALIFIED STRUCTURE NAMES
S	QUALIFIED SIRUCIORE MANES
CLASS L:	LITERALS
В	BIT STRING
C	CONVERSION TO INTERNAL FORMS
· F	FORMAT OF ARITHMETIC LITERALS
S	CHARACTER STRING
CLASS M:	MULTILINE FORMAT
	친 회사, 실용도 그들이 있는 것이 그의 남아 하다.
<u>C</u>	OVERPUNCH CONTEXT
E	E-LINE OVERPUNCH USE
0	

REPRODUCIBILITY OF THE ORIGINAL PAGE IS THE

s b	S-LINE COMMENTS
CLASS P:	PROGRAM CONTROL & INTERNAL CONSISTANCE
A C D E F L M P S T U b	ACCESS CONTROL COMPOOL BLOCKS DATA DEFINITION EXTERNAL TEMPLATES FUNCTION RETURN EXPRESSIONS LABELS MULTIPLE DEFINITIONS BLOCK DEFINITION PROCEDURE/FUNCTION TEMPLATES TASK DEFINITIONS CALLS FROM UPDATE BLOCKS MISCELLANEOUS
CLASS Q:	SHAPING FUNCTIONS
, A D S X	ARRAYNESS DIMENSION INFORMATION SUBSCRIPTS ARGUMENT TYPE
CLASS R:	REAL TIME STATEMENTS
E T U	ON/SEND ERROR STATEMENTS TIMING EXPRESSIONS UPDATE BLOCKS
CLASS S:	SUBSCRIPT USAGE
C P Q R S T V	SUBSCRIPT COUNT PUNCTUATION PRECISION QUALIFIER RANGE OF SUBSCRIPT VALUES USAGE OF ASTERISKS SUBSCRIPT TYPE VALIDITY OF USAGE

· CLASS T: I/O STATEMENTS

C CONTROL

D DEVICE NUMBER

b MISCELLANEOUS

ORIGINAL DES

CLASS U: UPDATE BLOCKS

I P T	IDENTIFIER USAGE PROGRAM BLOCKS I/O
CLASS V:	COMPILE-TIME EVALUATIONS
A C E F	ARITHMETIC OPERATIONS CATENATION OPERATIONS UNCOMPUTABLE EXPRESSIONS FUNCTION EVALUATION
CLASS X:	IMPLEMENTATION DEPENDENT FEATURES
A D I U	PROGRAM ID DIRECTIVE DEVICE DIRECTIVE INCLUDE DIRECTIVE UNKNOWN OR INVALID DIRECTIVE