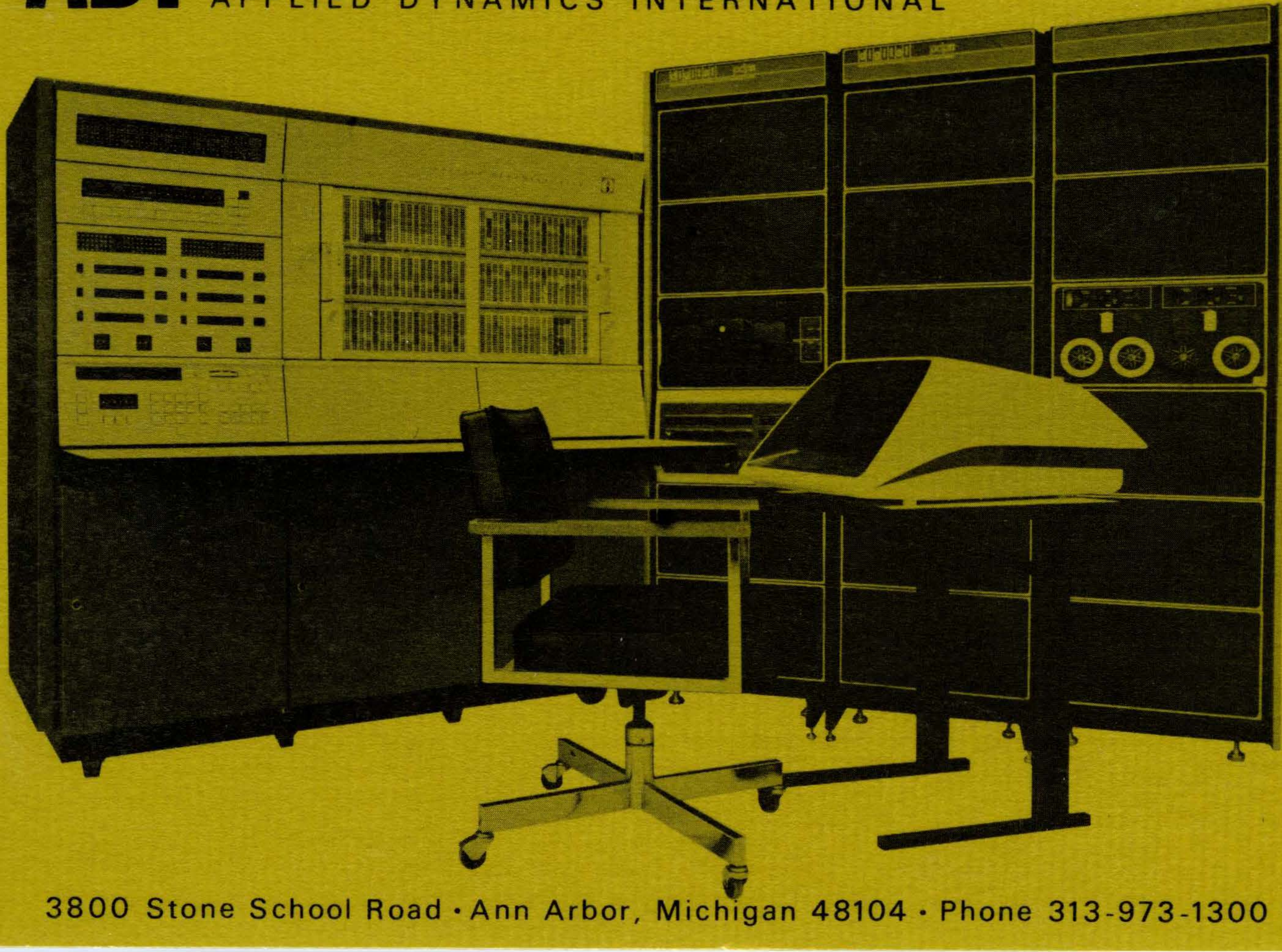
SYSTEM 511 USER'S MANUAL

INTERACTIVE HYBRID INTERPRETER

IHI

ADI APPLIED DYNAMICS INTERNATIONAL



3800 Stone School Road · Ann Arbor, Michigan 48104 · Phone 313-973-1300

IHI

INTERACTIVE HYBRID INTERPRETER

OPERATORS MANUAL

16 JAN 1976

APPLIED DYNAMICS INTERNATIONAL

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION TO IHI

IHI IS AN INTERACTIVE, EASY TO USE INTERPRETIVE PROGRAMMING LANGUAGE. ALTHOUGH IT IS A GENERAL PURPOSE COMPUTING LANGUAGE, ITS DESIGN PHILOSOPHY IS BASED ON THE NEEDS INHERENT IN A HYBRID COMPUTING ENVIRONMENT. IHI IS THE OUTGROWTH OF MANY YEARS OF EXPERIENCE AT APPLIED DYNAMICS, USING AND MODIFYING OTHER INTERPRETIVE PROGRAMMING LANGUAGES, SUCH AS DARTMOUTH BASIC, HYBRID BASIC (HYBASIC), INTERACTIVE HYBRID EXECUTIVE (IHE), FORTRAN INTERPRETER (FS), JUST TO NAME A FEW.

INTERACTIVE HYBRID COMMUNICATION CAPABILITY IS ESSENTIAL FOR EFFICIENT ANALOG AND HYBRID PROGRAM DEVELOPMENT AND DEBUGGING. IHI IS USEFUL IN A HYBRID SYSTEM IN MUCH THE SAME WAY THAT "ON-LINE-DEBUGGING" FEATURES ARE SOMETIMES INDISPENSABLE IN DIGITAL PROGRAM DEBUGGING. IHI IS DESIGNED TO RUN CONVENIENTLY ON BOTH SMALL AND LARGE (MULTI-CONSOLE) HYBRID SYSTEMS.

IHI HAS ALL THE NECESSARY STATEMENTS (VIA SUBROUTINE CALLS) TO "PUSH THE BUTTONS" ON THE ANALOG COMPUTER, AS WELL AS ALL THE BASIC HYBRID COMMUNICATION CAPABILITY PROVIDED BY THE HCR'S (HYBRID COMMUNICATION ROUTINES). IHI ALSO ALLOWS DIRECT ACCESS TO THE COMPONENTS IN THE ANALOG COMPUTER USING CONVENIENT ANALOG DEVICE MNEMONICS; FOR EXAMPLE, TO SET A COEFFICIENT DEVICE, THE USER CAN SIMPLY TYPE:

```
COF003 = .2367
```

OR TO READ OUT THE VALUE OF AN ANALOG COEFFICIENT AND USE IT IN AN EXPRESSION, THE USER MIGHT TYPE:

```
A=COF000/10
```

OR TO PRINT OUT THE VALUES OF SEVERAL ADC CHANNELS:

```
PRH ADC000,ADC001,ADC002,ADC003,ADC004,ADC005
```

MOST HYBRID COMMUNICATION ROUTINES (HCR'S) CAN BE CALLED FROM IHI. THE HCR CALLS FROM IHI ARE OF THE SAME FORM AS THE HCR CALLS FROM FORTRAN IV; THUS THE USER NEED NOT LEARN TWO LIBRARIES OF HYBRID COMMUNICATION ROUTINES. ALSO, AS AN ADDED FEATURE, SOME HCR'S WILL ACCEPT UNITY SCALED ARGUMENTS, WHICH ARE CONSISTENT WITH THE UNITY SCALING USED FOR THE ANALOG DEVICE MNEMONICS.

IHI HAS BOTH A "DEFERRED" AND AN "IMMEDIATE" MODE OF PROGRAM EXECUTION (MUCH LIKE "BASIC"). IN THE "IMMEDIATE" MODE, IHI STATEMENTS AND COMMANDS ARE ENTERED BY THE USER AND EXECUTED IMMEDIATELY. THIS MODE OF OPERATION IS USEFUL WHEN IHI IS USED AS A HYBRID DEBUGGING TOOL OR QUICK CALCULATOR. USUALLY PROGRAMS ARE PREPARED TO BE EXECUTED IN THE "DEFERRED" MODE. IN THIS CASE, EACH STATEMENT IS ASSIGNED A STATEMENT NUMBER WHICH DEFINES ITS POSITION IN THE EXECUTION OF THE PROGRAM.

THE USER CAN PREPARE AN IHI STORED PROGRAM "OFF-LINE" MUCH AS HE WOULD PREPARE A FORTRAN SOURCE PROGRAM, OR HE CAN ENTER, RUN, AND DEBUG A PROGRAM INTERACTIVELY WHILE RUNNING IHI. THE USER PROGRAM CAN BE INPUT FROM OR OUTPUT TO EITHER THE OPERATOR CONSOLE, A FILE ON MASS STORAGE, OR ANY APPROPRIATE I/O DEVICE.

IN GENERAL, THE EXECUTION OF MOST INTERPRETIVE LANGUAGE PROGRAMS IS NOT AS FAST AS THE EXECUTION OF COMPILER GENERATED OBJECT CODE. IHI HAS A PASS 1 PRE-PROCESSOR WHICH MINIMIZES THIS LIMITATION AND SPEEDS THE RUN TIME INTERPRETATION OF STORED PROGRAMS BY GENERATING A PARTIALLY INTERPRETED INTERNAL FORM OF THE SOURCE CODE.

THE I/O CAPABILITY OF IHI IS MUCH THE SAME AS FORTRAN IV STYLE I/O, WITH THE ADDITION OF SEVERAL FIXED FORMAT PRINT STATEMENTS. IHI PROVIDES FLEXIBLE INPUT AND OUTPUT OF BOTH PROGRAMS AND/OR COMPUTED DATA TO ANY APPROPRIATE PDP-11 I/O DEVICE. OUTPUT OF DATA CAN BE FORMATTED MUCH THE SAME WAY AS IN FORTRAN IV, AND INPUT DATA IN ANY FORMAT (INTEGER, REAL FLOATING, OR EXPONENTIAL) CAN BE READ USING ONE FLEXIBLE SPECIFIER; THE USER NEED NOT WORRY ABOUT DATA ITEMS BEING ENTERED IN A FIXED COLUMN, RIGHT JUSTIFIED, ETC., AS HE DOES IN FORTRAN IV.

THE VARIABLE NAMING CONVENTION IS ALSO CONVENIENT IN THAT IHI ALLOWS VARIABLE NAMES CONSISTING OF FROM 1 TO 6 ALPHA-NUMERIC CHARACTERS.

IHI PROGRAMS CAN BE COMMENTED BY SIMPLY PRECEEDING A COMMENT WITH A ";". A COMMENT CAN APPEAR ANYWHERE IN A STATEMENT LINE AND IHI WILL IGNORE THE REST OF THE LINE FOLLOWING THE ";". FOR EXAMPLE:

```
10.10 ;THIS STATEMENT IS JUST A COMMENT
10.20 A=1      ;ASSIGN A VALUE TO A
10.30 ;B=2     ;NOP A STATEMENT
```

TO IMPROVE READABILITY, IHI ALSO PERMITS (AND IGNORES) ANY NUMBER OF BLANKS OR TABS BETWEEN SYNTACTIC ENTITIES IN A STATEMENT.

IHI HAS MANY ADVANTAGES OVER A COMPILER ORIENTED LANGUAGE. IT PROVIDES A CONVENIENT MEANS OF ACHIEVING INITIAL FAMILIARITY WITH THE OPERATION OF A HYBRID COMPUTER SYSTEM. THIS IS ESPECIALLY IMPORTANT IN AN EDUCATIONAL ENVIRONMENT. FOR ON-LINE HYBRID DEBUGGING, IHI IS MUCH EASIER TO USE THAN WRITING, COMPILING, LINKING, INSTALLING, AND FINALLY EXECUTING A FORTRAN IV PROGRAM WHICH INTERROGATES THE ANALOG COMPUTER FOR PROBLEM DEBUGGING. IHI ALLOWS THE USER TO SIT AT THE PDP-11 OPERATOR CONSOLE, COMPOSE AND EXECUTE IMMEDIATELY A HYBRID COMPUTATIONAL ALGORITHM, AND SEE AT ONCE IF IT WORKS AS EXPECTED. ANY USER FAMILIAR WITH FORTRAN IV OR BASIC PROGRAMMING WILL FIND IHI EASY TO LEARN AND USEFUL IN MANY ANALOG/HYBRID APPLICATIONS.

TERMS AND CONVENTIONS USED IN THIS MANUAL

THIS SECTION DEFINES A NUMBER OF TERMS AND CONVENTIONS USED IN THIS MANUAL.

RSX-11 — REAL TIME OPERATING SYSTEM FOR THE PDP-11

MCR> — RSX-11 MONITOR CONSOLE ROUTINE PROMPT

LUN — LOGICAL UNIT NUMBER

< > — THESE SYMBOLS ARE USUALLY USED TO DELIMIT THE NAME OF A SYNTACTIC ELEMENT, OR A CLASS OF ELEMENTS, SUCH AS:

```
        <VARIABLE>
        <VARIABLE LIST>
        <CONTROL Z>
```

[ ] — BRACKETS ARE USED TO DELIMIT OPTIONAL PORTIONS OF A SYNTACTIC DEFINITION.

GG.LL — INDICATES AN IHI STATEMENT NUMBER, WHERE "GG" (1-99) IS THE GROUP NUMBER AND "LL" (1-99) IS THE LINE NUMBER. IN SOME CASES THE ".LL" IS OPTIONAL AND THE ENTIRE GROUP IS REFERENCED. ALSO, IN MOST CONTEXTS, THE STATEMENT NUMBER CAN BE INDICATED AS A VARIABLE OR ARITHMETIC EXPRESSION. THE INTEGER PORTION OF THE REAL VALUE IS THE GROUP (GG) AND THE FRACTIONAL PART IS THE LINE NUMBER (LL).

RAD50 — INTERNALLY, IHI STORES ASCII NAMES IN RADIX 50 PACKED FORM. THE VALID RAD50 CHARACTERS ARE: <BLANK>, A-Z, 0-9, ".", AND "$".

VERB — "VERB" REFERS TO THE INITIAL SYNTACTIC ELEMENT IN A COMMAND OR STATEMENT, I.E. "RUN", "LIST" "READ", "FORMAT", ETC.

BLANKS & TABS — THE USER CAN FREELY INSERT ANY NUMBER OF BLANKS AND/OR TABS BETWEEN SYNTACTIC ELEMENTS IN IHI COMMANDS AND STATEMENTS. HOWEVER, BLANKS AND/OR TABS CAN NOT BE INSERTED WITHIN A SYNTACTIC ENTITY, SUCH AS A VARIABLE NAME, STATEMENT "VERB", NUMERIC CONSTANT, ETC. THERE ARE SOME INSTANCES WHERE A NON-RAD50 CHARACTER OR A BLANK OR TAB IS MANDATORY, FOR EXAMPLE, FOLLOWING MOST IHI STATEMENT "VERBS". IN THESE CASES, A BLANK IS SHOWN IN THE STATEMENT SYNTAX.

IMMEDIATE MODE IN THE "IMMEDIATE" MODE OF EXECUTION ANY VALID IHI COMMAND OR STATEMENT ENTERED IN RESPONSE TO THE "IMMEDIATE" MODE PROMPT "?" IS EXECUTED IMMEDIATELY. IF A STATEMENT IS PRECEDED BY A STATEMENT NUMBER (GG.LL), THE STATEMENT IS STORED FOR EXECUTION IN THE "DEFERRED" MODE.

DEFERRED MODE STATEMENTS PRECEDED BY STATEMENT NUMBERS ARE STORED FOR "DEFERRED" MODE EXECUTION. WHEN THE STORED STATEMENTS ARE EXECUTED IN THE "DEFERRED" MODE, THE EXECUTION PROCEEDS ACCORDING TO STATEMENT NUMBER, FROM THE SMALLEST TO THE LARGEST, OR THE PROGRAM FLOW IS DETERMINED BY THE STATEMENT EXECUTION ("JUMP", "GOSUB", ETC.)

PROMPT MODE THE PROMPT MODE IS ENTERED BY TYPING "PRMT VERB" IN RESPONCE TO IHI'S "?". THEREAFTER, IHI WILL PROMPT WITH: "?VERB ", AND THE USER MAY TYPE THE REMAINDER IF THE STATEMENT. IHI WILL INTERPRET THE STATEMENT THE SAME AS IF THE USER HAD TYPED THE "VERB ". THE USER MAY EXIT FROM THE PROMPT MODE BY TYPING A CARRIAGE RETURN IN RESPONCE TO "VERB ". ONLY A SUBSET OF THE IHI COMMAND AND STATEMENT VERBS ARE LEGAL (SEE APPENDIX A).

DIFFERENCES UNDER RSX-11D AND RSX-11M

1) UNDER RSX-11D <CONTROL X> IS USED TO INTERRUPT A RUNNING DEFERRED MODE PROGRAM BY RUNNING THE TASK "TTYNXX", WHERE "XX" IS THE TWO DIGIT OCTAL TERMINAL NUMBER.

   UNDER RSX-11M INTERRUPT VIA <CONTROL X> IS NOT SUPPORTED. INSTEAD OF <CONTROL X>, THE USER TYPES "PUNT" OR "PUN" TO INTERRUPT A RUNNING DEFERRED MODE PROGRAM. THE UNSOLICITED INPUT GOES TO "MCR", WHICH RUNS THE TASK "...PUN".

2) UNDER RSX-11M THE ERROR MESSAGE FILE "IHIINS.MSG" DESCRIBED IN CHAPTER 8 IS NOT USED. THE EQUIVALENT MESSAGES ARE STORED IN MEMORY AND PRINTED WHEN NECESSARY.

CHAPTER 2
LOADING AND RUNNING IHI

THE INTERACTIVE HYBRID INTERPRETER IS RUN MUCH AS ANY OTHER RSX-11 SYSTEM PROGRAM BY TYPING (IN RESPONSE TO THE MCR PROMPT):

```
        MCR>IHI [<COMMAND STRING>]
```

IF THE OPTIONAL <COMMAND STRING> IS NOT ENTERED, THEN IHI PROMPTS AS FOLLOWS, REQUESTING A <COMMAND STRING> :

```
        IHI>
```

THE <COMMAND STRING> FORMAT AND MEANING IS DEFINED AS FOLLOWS:

```
                @<FILE SPECIFICATION>
        OR      /SW/SW/SW ...
```

WHERE THE FIRST FORM INDICATES THAT COMMANDS TO IHI WILL COME FROM THE FILE INDICATED IN THE INDIRECT <FILE SPECIFICATION>, AND THE SECOND FORM INDICATES A LIST OF SWITCHES WHICH INITIALIZE OR COMMAND THE IHI PROCESSOR.

NOTE: IF NEITHER FORM IS DESIRED, THE USER CAN SIMPLY TYPE <CARRIAGE RETURN>.

THE FIRST FORM OF THE <COMMAND STRING> INDICATES AN INDIRECT FILE WHICH WILL CONTAIN AN OPTIONAL <COMMAND STRING> IN EITHER FORM, FOLLOWED BY ANY NUMBER OF "IMMEDIATE" MODE COMMANDS AND/OR "DEFERRED" MODE IHI STATEMENTS (PRECEDED BY A STATEMENT NUMBER). REFER TO THE NEXT SECTION OF THIS CHAPTER FOR FURTHER DISCUSSION OF INDIRECT COMMAND FILES. FOLLOWING THE EXECUTION OF THE COMMANDS IN THE INDIRECT COMMAND FILE, EITHER MCR RETURNS, IF THE INDIRECT COMMAND FILE SPECIFICATION IS ENTERED ON THE SAME LINE AS THE MCR PROMPT:

```
                MCR>IHI @FILE
                MCR>
```

OR IHI RETURNS TO THE "IMMEDIATE" MODE AND PRINTS ITS PROMPT "?" IF THE INDIRECT COMMAND FILE SPECIFICATION IS ENTERED FOLLOWING THE IHI PROMPT "IHI>":

```
                MCR>IHI
                IHI>@FILE
                ?
```

IF THE INDIRECT COMMAND FILE SPECIFICATION IS ENTERED IN RESPONSE TO "IHI>" OR AS FOLLOWS: "MCR>IHI @FILE", THEN THE FIRST RECORD IN THE INDIRECT FILE MUST BE A VALID <COMMAND STRING> AS DEFINED IN CHAPTER 2, OR A BLANK RECORD TERMINATED BY A <CARRIAGE RETURN>.

THE SECOND FORM OF THE COMMAND STRING (SW/SW/SW ...) IS A LIST OF SWITCHES DEFINED AS FOLLOWS:

SWITCH CODE	MEANING
C0	ATTACH TO ANALOG CONSOLE 0 AND ADC 0
C1	ATTACH TO ANALOG CONSOLE 1 AND ADC 1
ID	PRINT IHI VERSION NUMBER AND POOL SIZE, IGNORE ANY OTHER SWITCHES, AND ISSUE ANOTHER PROMPT "IHI>".

THE SECOND FORM OF THE <COMMAND STRING> IS NOT VERY USEFUL IF ENTERED ON THE SAME LINE AS THE MCR PROMPT ("MCR>") I.E.

```
MCR>IHI /C0
MCR>
```

SINCE AS ILLUSTRATED, IHI ATTACHES TO CONSOLE 0 AND IMMEDIATELY EXITS AND MCR RETURNS.

ONCE IHI HAS INTERPRETED THE INPUT <COMMAND STRING> AND EXECUTED ANY IMMEDIATE MODE COMMANDS, IN THE CASE OF AN INDIRECT COMMAND FILE, IT RESPONDS WITH ITS "IMMEDIATE" MODE REQUEST FOR INPUT PROMPT:

```
?
```

THERE ARE TWO MODES OF OPERATION OF IHI: "IMMEDIATE MODE" AND "DEFERRED MODE". IN THE "IMMEDIATE" MODE, IHI WILL IMMEDIATELY EXECUTE THE STATEMENT ENTERED IN RESPONSE TO THE "?". THERE ARE A NUMBER OF "IMMEDIATE"-MODE-ONLY IHI COMMANDS WHICH IMPLEMENT IHI CONTROL AND COMMAND FUNCTIONS. IN THE "DEFERRED" MODE OF OPERATION, STATEMENTS PRECEDED BY A STATEMENT NUMBER OF THE FORM:

```
GG.LL

WHERE "GG" IS THE GROUP NUMBER (1 TO 99)
AND "LL" IS THE LINE NUMBER (1 TO 99)
```

ARE STORED IN MEMORY AND EXECUTED AT A LATER TIME. "DEFERRED" MODE STATEMENTS CAN BE ENTERED IN RESPONSE TO THE "?" OR FROM A FILE ON A MASS STORAGE DEVICE (USING AN INDIRECT COMMAND FILE SPECIFICATION).

TO ABORT IHI AND RETURN TO "MCR>", TYPE <CONTROL Z>. TO INTERRUPT A RUNNING "DEFERRED" MODE PROGRAM AND RETURN TO THE IHI "IMMEDIATE" MODE, FOLLOWING THE EXECUTION OF THE CURRENT STATEMENT, TYPE <CONTROL X>.

IHI IMMEDIATE MODE COMMANDS

THE FOLLOWING IS A DESCRIPTION OF THE IHI "IMMEDIATE" MODE COMMANDS. MOST IHI LANGUAGE STATEMENTS (SEE CHAPTER 7) CAN ALSO BE EXECUTED IN THE "IMMEDIATE" MODE.

INDIRECT COMMAND FILE SPECIFICATION

```
@<FILE SPECIFICATION>
```

ANYTHING THAT CAN BE ENTERED IN RESPONSE TO "IHI>" OR IHI'S REQUEST FOR INPUT PROMPT ("?"), CAN ALSO BE STORED IN AN ASCII FILE AND ENTERED INDIRECTLY USING THE INDIRECT COMMAND FILE SPECIFICATION. THIS APPLIES TO "IMMEDIATE" MODE COMMANDS DISCUSSED IN THIS SECTION, "DEFERRED" MODE STATEMENTS (PRECEDED BY A STATEMENT NUMBER), OR A COMBINATION OF BOTH.

AN INDIRECT COMMAND FILE MAY CONTAIN ANOTHER INDIRECT FILE SPECIFICATION. HOWEVER, THE MAXIMUM INDIRECT FILE NESTING LIMIT IS 3 (I.E. UP TO 3 INDIRECT COMMAND FILES CAN BE OPEN AT ONE TIME).

THE FOLLOWING EXAMPLE OF AN INDIRECT COMMAND FILE CONTAINS A "DEFERRED" MODE PROGRAM AND THE COMMANDS TO LIST IT ON THE LINE PRINTER AND RUN IT:

```
10.10 ;PRINT A TABLE OF THE SINE AND COSINE FUNCTIONS
10.20 FOR I=0,2*3.14159,.1
10.22 SINE=SIN(I)
10.24 COSINE=COS(I)
10.30 WRITE(1,10.40)I,SINE,COSINE
10.40 FORMAT(1X(5X,E14.7))
10.50 NEXT I
10.60 STOP

LIST 'LP:'
RUN
```

ASSUMING THE ABOVE EXAMPLE IS CONTAINED IN "DK0:FILE.CMD;2", THEN THE USER CAN ENTER, LIST, AND RUN THE PROGRAM BY SIMPLY TYPING IN RESPONSE TO ("?"):

```
?@DK0:FILE.CMD;2
```

THE DEFAULT DEVICE IS "SY:", THE DEFAULT EXTENSION IS ".CMD", AND IHI LOOKS FOR THE MOST RECENT VERSION IF NONE IS SPECIFIED. THE FOLLOWING WOULD BE EQUIVALENT TO THE PREVIOUS SPECIFICATION (ASSUMING VERSION 2 WAS THE MOST RECENT):

```
?@FILE
```

RUN COMMAND

```
RUN
```

THE "RUN" COMMAND INITIALIZES ALL CORE BLOCKS ASSOCIATED WITH RUNNING AN IHI PROGRAM AND THEN BEGINS EXECUTION OF THE CURRENT STORED PROGRAM, BEGINNING WITH THE STORED STATEMENT WITH THE SMALLEST STATEMENT NUMBER.

THE FOLLOWING EXAMPLE ILLUSTRATES THE ENTRY AND RUNNING OF A SIMPLE "DEFERRED" MODE PROGRAM:

```
?10.10 ANGLE = 6.28*(47.2/360)    ;47.2 DEGREES IN RADIANS
?10.20 PRE SIN(ANGLE),COS(ANGLE),SIN(ANGLE)/COS(ANGLE)
?RUN
SIN(ANGLE) = 0.7334E 00
COS(ANGLE) = 0.6797E 00
SIN(ANGLE)/COS(ANGLE) = 0.1079E 01
IHI --WARN-- STOP AT LINE 10.20

?
```

WARNING: IF A STORED MODE PROGRAM IS ENTERED FOR THE FIRST TIME FROM THE CONSOLE IN THE "IMMEDIATE" MODE, IT IS A GOOD IDEA TO USE THE "LIST" OR "SAVE" COMMAND TO PRESERVE A COPY OF THE PROGRAM PRIOR TO ISSUING A "RUN" COMMAND. THIS GIVES A BACKUP COPY WHICH CAN BE USED TO RESTORE THE PROGRAM IN CASE IHI MUST BE ABORTED FOR SOME REASON.

THE USER CAN INTERRUPT A RUNNING IHI PROGRAM BY TYPING <CONTROL X>. IHI WILL RESPOND WHEN IT COMPLETES EXECUTION OF THE CURRENT STATEMENT AND RETURN TO THE "IMMEDIATE" MODE. THIS IS ESPECIALLY USEFUL IF A PROGRAM GETS HUNG IN AN INFINITE LOOP.

LIST COMMAND

```
LIST ['<FILE SPECIFICATIION>'] [ GG[.LL] [,GG[.LL]] ]

WHERE OPTIONAL ITEMS ARE DELIMITED BY []'S
```

THE "LIST" COMMAND CONVERTS THE STORED IHI PROGRAM FROM ITS INTERNAL FORM (PACKED BINARY CODE) TO ASCII AND LISTS IT TO THE <FILE SPECIFICATION> OR TO THE CONSOLE ("TI:IHISAV.CMD") IF NO FILE IS INDICATED. IF THE STATEMENT NUMBER PAIR IS INDICATED, THEN IHI LISTS FROM THE FIRST GG.LL TO THE SECOND GG.LL (INCLUSIVE). IF ONLY THE FIRST GG.LL IS INDICATED, THEN ONLY THAT LINE NUMBER ("GG.LL") OR GROUP ("GG") IS LISTED. IF NO STATEMENT NUMBERS ARE INDICATED, THEN THE ENTIRE STORED PROGRAM IS LISTED (EQUIVALENT TO "LIST '<FILE SPECIFICATION>'1,99.99").

TO LIST THE ENTIRE STORED PROGRAM ON THE LINE PRINTER THE USER COULD TYPE:

```
LIST 'LP:'
```

TO LIST ALL STATEMENTS IN GROUP 23 TO A DISK FILE "IHI.TMP" THE USER COULD TYPE:

```
LIST 'DK1:IHI.TMP' 23
```

OR TO LIST JUST LINE 83.78 ON THE OPERATOR CONSOLE THE USER WOULD TYPE:

```
LIST 83.78
```

THE FOLLOWING LISTS ALL STATEMENTS FROM LINE 10.20 UP TO GROUP 30 TO THE DEFAULT FILE NAME "IHISAV.CMD", SINCE ONLY "DK1:" IS SPECIFIED:

```
LIST 'DK1:' 10.20,30     ;"30" DEFAULTS TO LINE "30.99"
                         ;(ALL OF GROUP 30 IS LISTED)
```

THE FOLLOWING EXAMPLE ILLUSTRATES THE USE OF THE "LIST" COMMAND:

```
?10.10 A=1
?10.30 ZZZZZZ=A**Z
?10.20 Z=SQR(A)
?10.10 A=1.11111
?LIST

10.10  A=1.11111
10.20  Z=SQR(A)
10.30  ZZZZZZ=A**Z

?LIST 10.10,10.20

10.10  A=1.11111
10.20  Z=SQR(A)
```

(NOTE: "LIST" IS ALSO A LEGAL "DEFERRED MODE" STATEMENT)

DELETE COMMAND

```
GG.LL
DELETE GG[.LL] [,GG[.LL]]
```

WHERE OPTIONAL ITEMS ARE DELIMITED BY []'S

THE "DELETE" COMMAND IS USED TO DELETE ONE OR MORE STORED IHI STATEMENTS. THE FIRST FORM, A STATEMENT NUMBER FOLLOWED BY A <CARRIAGE RETURN>, IS A DEFAULT DELETE AND WILL DELETE THAT STATEMENT IF IT EXISTS.

THE SECOND FORM WILL DELETE ALL STATEMENTS FROM THE FIRST "GG.LL" THROUGH THE SECOND GG.LL. IF THE LINE NUMBER ".LL" IS NOT INDICATED, THEN ALL STATEMENTS FROM THE FIRST GROUP "GG" THROUGH THE SECOND GROUP "GG" ARE DELETED. ONLY THE FIRST OF THE OF THE PAIR OF STATEMENT NUMBERS NEED BE SPECIFIED, IN WHICH CASE ONLY THAT LINE (GG.LL) OR GROUP (GG) IS DELETED.

TO DELETE ALL STATEMENTS WITH GROUP NUMBER "10" THE USER COULD TYPE:

```
        DELETE 10       ;EQUIVALENT TO "DELETE 10.00,10.99"
```

THE FOLLOWING WOULD DELETE JUST STATEMENT 10.10

```
        10.10
OR      DELETE 10.10
```

AND TO DELETE ALL STATEMENTS IN ALL GROUPS THE USER COULD TYPE:

```
        DELETE 1,99
```

THE FOLLOWING:

```
        DELETE 10.35,50        ;"50" DEFAULTS TO LINE "50.99"
```

WOULD DELETE ALL STATEMENTS FROM LINE 10.35 THROUGH GROUP 50 (ALL STATEMENTS IN GROUP 50 WILL BE DELETED).

ZAP COMMAND

```
ZAP
```

THE "ZAP" STATEMENT DELETES ALL CORE BLOCKS THAT WERE ALLOCATED EITHER BY THE RUNNING OF A "DEFERRED" MODE PROGRAM OR BY THE EXECUTION OF "IMMEDIATE" MODE IHI STATEMENTS. THIS INCLUDES:

CALL BLOCKS
FOR-NEXT BLOCKS
PAUSE-CON BLOCKS
TEMPORARY BLOCKS USED BY IHI
SYMBOL TABLE (VARIABLES AND ARRAYS)

"ZAP" DOES NOT DELETE THE STORED IHI STATEMENTS (SEE "RENEW" COMMAND). IN GENERAL, "ZAP" DOES THE SAME THING THAT THE "RUN" COMMAND DOES PRIOR TO RUNNING A "DEFERRED" MODE PROGRAM. A "ZAP" FOLLOWED BY A "JUMP" TO THE FIRST STATEMENT IN A PROGRAM IS EQUIVALENT TO A "RUN" COMMAND.

RENEW COMMAND

```
RENEW
```

THE "RENEW" COMMAND DELETES ALL STORED STATEMENTS AND INITIALIZES ALL CORE BLOCKS WHICH HAVE BEEN ALLOCATED. THE "RENEW" COMMAND IS EQUIVALENT TO A "DELETE 1,99.99" FOLLOWED BY A "ZAP" COMMAND.

SAVE COMMAND

```
SAVE ['<FILE SPECIFICATION>'] [ GG[.LL] [,GG[.LL]] ]

WHERE OPTIONAL ITEMS ARE DELIMITED BY [ ]'S
```

THE "SAVE" COMMAND CAUSES THE PACKED BINARY CORE IMAGE OF THE STORED IHI STATEMENTS FROM THE FIRST GG.LL THROUGH SECOND GG.LL TO BE DUMPED TO THE <FILE SPECIFICATION>. IF NO FILE IS SPECIFIED, THE STATEMENTS ARE DUMPED TO "SY:IHISAV.IHI". A NEW VERSION IS CREATED IF THE FILE NAME SPECIFIED ALREADY EXISTS. IF NO STATEMENT NUMBER PAIR IS SPECIFIED, THEN ALL STORED STATEMENTS ARE SAVED. IF ONLY THE FIRST NUMBER IN THE PAIR IS SPECIFIED, THEN THE STATEMENT (GG.LL) OR THE GROUP (GG) IS SAVED. THE "LOAD" COMMAND CAN BE USED TO RE-LOAD A STORED PROGRAM THAT WAS SAVED USING THE "SAVE" COMMAND.

THE "LIST" COMMAND CAN BE USED IF THE USER WISHES TO SAVE A SOURCE CODE REPRESENTATION OF A STORED IHI PROGRAM; A SOURCE CODE REPRESENTATION OF A PROGRAM CAN BE RE-LOADED USING THE INDIRECT COMMAND FILE SPECIFICATION.

THE FOLLOWING EXAMPLE ILLUSTRATES SEVERAL VALID FORMS OF THE "SAVE" COMMAND:

```
SAVE                ;TO "SY:IHISAV.IHI"
SAVE 'DK1:'         ;SAVE ALL STATEMENTS TO "DK1:IHISAV.IHI"
SAVE 'SAVE.IHI'     ;SAVE ALL STATEMENTS TO "SY:SAVE.IHI"
SAVE 'SAVE' 10      ;SAVE GROUP 10 TO "SY:SAVE.IHI"
SAVE 10.10,10.20    ;SAVE FROM LINE 10.10 THROUGH 10.20
SAVE 10,20          ;SAVE GROUPS 10 THROUGH 20 (INCLUSIVE)
SAVE 10,21          ;EQUIVALENT TO "SAVE 10,21.99"
SAVE ';2'           ;SAVE TO "SY:IHISAV.IHI;2"
```

LOAD COMMAND

```
LOAD ['<FILE SPECIFICATION>']
```

THIS STATEMENT CAUSES THE PACKED BINARY CORE IMAGE OF THE IHI STATEMENTS WHICH WERE SAVED VIA THE "SAVE" STATEMENT TO BE LOADED. THE CURRENT STORED PROGRAM IS DELETED PRIOR TO THE LOADING PROCESS. THE DEFAULT LOAD FILE IS THE MOST RECENT VERSION OF 'SY:IHISAV.IHI'. ANY PORTION OF THE <FILE SPECIFICATION> NOT SPECIFIED WILL DEFAULT TO THE CORRESPONDING PART OF THE DEFAULT <FILE SPECIFICATION>.

A SOURCE CODE REPRESENTATION OF A DEFERRED MODE IHI PROGRAM CAN BE LOADED BY STORING IT IN A FILE AND USING THE INDIRECT COMMAND FILE SPECIFICATION AS DISCUSSED EARLIER IN THIS CHAPTER.

THE FOLLOWING ARE SOME EXAMPLES OF THE "LOAD" COMMAND:

```
LOAD                ;LOAD FROM MOST RECENT "SY:IHISAV.IHI"
LOAD 'SAVE'         ;LOAD MOST RECENT VERSION OF "SY:SAVE.IHI"
LOAD 'DK1:FILE.EXT;2'
LOAD 'DK1:'         ;LOAD FROM MOST RECENT "DK1:IHISAV.IHI"
LOAD ';5'           ;LOAD FROM "SY:IHISAV.IHI;5"
```

OVRLAY COMMAND

OVRLAY ['<FILE SPECIFICATION>']

THE "OVRLAY" COMMAND FUNCTIONS LIKE "LOAD" EXCEPT THAT THE CURRENT STORED PROGRAM IS NOT DELETED PRIOR TO THE LOADING PROCESS. STATEMENTS WHICH ALREADY EXIST WILL BE SUPERSEDED BY THE INCOMING STATEMENTS IN THE OVERLAY FILE. THE DEFAULT FILE IS "SY:IHISAV.IHI" AND <FILE SPECIFICATION> CAN BE USED TO MODIFY ALL OR PART OF THIS DEFAULT NAME.

THE FOLLOWING EXAMPLE ILLUSTRATES THE USE OF THE "OVRLAY" COMMAND:

```
?10.10 WRITE(0,10.20)
?10.20 FORMAT(' MAIN PROGRAM')
?SAVE 'MAIN'
?RENEW
?10.20 FORMAT(' SEGMENT 1')
?SAVE 'SEG1'
?RENEW
?10.20 FORMAT(' SEGMENT 2')
?SAVE 'SEG2'
?LOAD 'MAIN'
?RUN
MAIN PROGRAM
IHI --WARN-- STOP AT LINE 10.20

?OVRLAY 'SEG1'
?RUN
SEGMENT 1
IHI --WARN-- STOP AT LINE 10.20

?OVRLAY 'SEG2'
?RUN
SEGMENT 2
IHI --WARN-- STOP AT LINE 10.20

?
```

CONTINUE COMMAND

CON

THE "CON" COMMAND BECOMES A LEGAL "IMMEDIATE" MODE COMMAND FOLLOWING THE EXECUTION OF A "DEFERRED" MODE "PAUSE" STATEMENT (SEE CHAPTER 7), AND CONTINUES EXECUTION OF THE DEFERRED MODE PROGRAM AT THE STATEMENT FOLLOWING THE LAST EXECUTED "PAUSE" STATEMENT.

IN THE NORMAL SITUATION THE USER WILL PROBABLY TYPE ONE "CON" FOR EACH "PAUSE" STATEMENT. HOWEVER IF HE CONTINUES THE DEFERRED MODE PROGRAM USING SOME OTHER STATEMENT (FOR EXAMPLE "JUMP" OR "GOSUB"), IHI HAS STACKED THE CONTINUE ADDRESS FOR THE "CON" STATEMENT AND "CON" IS STILL VALID. IF ANOTHER "PAUSE" GETS EXECUTED, THEN "CON" WILL BEHAVE NORMALLY AND CONTINUE FOLLOWING THAT "PAUSE". HOWEVER, THE PREVIOUS "CON" HAS NOT BEEN FORGOTTEN AS THE FOLLOWING EXAMPLE ILLUSTRATES:

```
?10.10 PAUSE
?10.11 STOP
?10.12 PAUSE
?10.13 STOP

?RUN
IHI --WARN-- PAUSE AT LINE 10.10

?JUMP 10.12
IHI --WARN-- PAUSE AT LINE 10.12

?CON
IHI --WARN-- STOP AT LINE 10.13

?CON
IHI --WARN-- STOP AT LINE 10.11

?
```

PROMPT ("PRMT") COMMAND

PRMT <VERB>

THE "PRMT" COMMAND ALLOWS THE USER TO ENTER AN IHI COMMAND OR STATEMENT <VERB> WHICH THEREAFTER WILL BE APPENDED TO IHI'S NORMAL PROMPT "?". WHEN IHI PROMPTS WITH "?VERB ", THE USER MAY TYPE THE REMAINDER OF THE STATEMENT AND IHI WILL INTERPRET IT JUST AS IF THE USER HAD TYPED THE PROMPTING "VERB ". THE USER MAY EXIT FROM THE PROMPT MODE BY TYPING JUST A CARRIAGE RETURN, AND IHI WILL PROMPT WITH ITS NORMAL "?". ONLY THE SUBSET OF IHI <VERB>S INDICATED IN APPENDIX A ARE LEGAL IN THE PROMPT MODE. THE FOLLOWING EXAMPLE ILLUSTRATES USE OF "PRMT":

```
?PRMT DELETE
?DELETE 10,20            ;USER TYPED "10,20"
?DELETE 30.20,30.30      ;USER TYPED "30.20,30.30"
?DELETE 40,50.10         ;USER TYPED "40,50.10"
?DELETE                  ;USER TYPED <CARRIAGE RETURN>
?
```

CHAPTER 3
CONSTANTS, VARIABLES, AND ARRAYS

IN IHI ALL CONSTANTS AND VARIABLES ARE STORED INTERNALLY AS TWO WORD SINGLE PRECISION REAL VALUES. THIS YIELDS ROUGHLY 7 SIGNIFICANT DIGITS OF ACCURACY OVER THE REAL NUMBER RANGE FROM APPROXIMATELY .28E-38 TO 1.7E+38 . ONE AND TWO DIMENSIONED ARRAYS ARE ALLOWED AND ARE DIMENSIONED USING THE IHI "DIM" STATEMENT (SEE CHAPTER 7). EACH ELEMENT IN AN ARRAY IS ALSO A SINGLE PRECISION REAL VALUE.

NUMERIC CONSTANTS

A CONSTANT CAN BE REPRESENTED IN INTEGER FORMAT, FOR EXAMPLE:

```
10000
0
```

OR AS A REAL NUMBER WITH A DECIMAL POINT:

```
0.0023
0.0
```

OR IN EXPONENTIAL NOTATION (WHERE THE "E" FIELD INDICATES MULTIPLICATION BY A POWER OF 10):

```
.1E10          (.1 * (10**10) )
0.2345E-2      (.2345 * (10**(-2)) )
1E+2           (1.0 * (10**2))  )
```

THERE IS NO LIMIT TO THE NUMBER OF DIGITS THAT CAN BE SPECIFIED. HOWEVER, WHEN THE CONSTANT IS EVALUATED THE RESULT WILL STILL ONLY HAVE ABOUT 7 SIGNIFICANT DIGITS OF ACCURACY, AS THE FOLLOWING EXAMPLE ILLUSTRATES:

```
?10.10  A=1234567890
?10.20  WRITE(0,10.30)A
?10.30  FORMAT(1X,E20.13)
?RUN
0.1234567910432E 10
```

THE FOLLOWING ARE SOME EXAMPLES OF INVALID CONSTANTS:

```
E10     (AT LEAST ONE DIGIT MUST PRECEED "E")
.E-2    (        "       "       "      "   )
1.3E    (AT LEAST ONE DIGIT MUST FOLLOW "E")
1.2E.1  (EXPONENT MUST BE AN INTEGER)
1E 10   (SPACES NOT ALLOWED WITHIN CONSTANT)
```

STRING CONSTANTS

A STRING CONSTANT CONSISTS OF FROM 1 TO 4 CHARACTERS AND IS STORED LEFT JUSTIFIED IN TWO WORDS AS A REAL VALUE. CHARACTER STRINGS ARE DELIMITED BY SINGLE QUOTES ('); STRINGS LONGER THAT 4 CHARACTERS ARE TRUNCATED TO 4 CHARACTERS AND STRINGS SHORTER THAN 4 CHARACTERS ARE PADDED WITH TRAILING BLANKS. A SINGLE QUOTE CAN BE INCLUDED IN A STRING BY ENTERING TWO ADJACENT SINGLE QUOTES AT THE POINT WHERE ONE IS DESIRED IN THE CHARACTER STRING. THE FOLLOWING ARE EXAMPLES OF VALID STRING CONSTANTS:

```
'ABCD'    IS THE STRING: ABCD
'A  B'    IS THE STRING: A  B
'ABCDEF'  IS THE STRING: ABCD
''''      IS THE STRING: '
'"'       IS THE STRING: "
```

AND THE FOLLOWING ARE EXAMPLES OF INVALID STRING CONSTANTS:

```
'ABC            (MISSING TRAILING DELIMITER)
"++?."          (" CANNOT BE USED TO DELIMIT A STRING)
''              (A NULL STRING IS NOT ALLOWED)
'               (THIS HAS EVEN LESS MEANING THAN A NULL STRING)
```

VARIABLES

ORDINARY VARIABLES CAN BE NAMED USING FROM ONE TO SIX ALPHANUMERIC CHARACTERS, STARTING WITH AN ALPHABETIC CHARACTER (AS IN FORTRAN IV). THE FOLLOWING ARE LEGAL VARIABLE NAMES:

```
X
NUMBER
A00000
L0D5P
```

AND THE FOLLOWING ARE ILLEGAL VARIABLE NAMES:

100A	(BEGINS WITH A NUMERIC CHARACTER)
$$$JLP	(BEGINS WITH "$")
D.J.E.	(CONTAINS INVALID "."S)
ERDVILAS	(TOO LONG)

IN GENERAL, THERE ARE NO RESTRICTED NAMES. HOWEVER, THERE ARE TWO CLASSES OF NAMES, "ANALOG DEVICE MNEMONICS" AND "ARITHMETIC FUNCTIONS", WHICH PERFORM PRE-DEFINED FUNCTIONS IN IHI. USE OF THESE AS VARIABLE NAMES INVALIDATES THE OPERATION OF THESE FUNCTIONS.

ANALOG DEVICE MNEMONIC NAMES ARE OF THE FORM:

```
DEVNNN
```

WHERE "DEV" IS ONE OF THE ANALOG DEVICE CODES AND "NNN" IS AN ANALOG ADDRESS IN THE RANGE 0-59 OR 100-159

AND ARE NORMALLY USED TO ACCESS THE ANALOG COMPUTER(S). IF A "READABLE" ANALOG DEVICE MNEMONIC APPEARS TO THE LEFT OF AN "=", THEN IT BECOMES AN ORDINARY VARIABLE.

PRE-DEFINED ARITHMETIC FUNCTION NAMES (SEE CHAPTER 5) CANNOT BE USED SIMULTANEOUSLY AS BOTH VARIABLES AND FUNCTIONS. THE REASON IS THAT ONCE A VARIABLE IS CREATED WITH THE SAME NAME AS AN ARITHMETIC FUNCTION, THAT FUNCTION CAN NO LONGER BE USED. FOR EXAMPLE, THE FOLLOWING CREATES THE VARIABLE "SIN" WITH THE VALUE 1.3:

```
SIN=1.3
```

AND THE FOLLOWING WOULD CAUSE AN ERROR MESSAGE (SINCE "SIN" IS NOW A VARIABLE):

```
SINE=SIN(3.14159)
```

ALL IHI STATEMENT "VERBS" (SUCH AS "LET", "WHEN", "READ", "PRE", ETC.) CAN BE USED AS VARIABLE NAMES WITH NO CONFLICT.

A VARIABLE IS CREATED BY ASSIGNING IT A VALUE, I.E.

```
                A=1     ;CREATES VARIABLE "A" WITH VALUE 1.0
```

OR BY READING IN A VALUE FOR IT:

```
                READ(0,10.19)A
```

OR BY DIMENSIONING IT:

```
                DIM A(0,0)
```

NOTE: TO USE "VIR" AS AN ORDINARY VARIABLE, A SYMBOL TABLE ENTRY MUST BE CREATED EITHER VIA A "DIM" OR "READ" STATEMENT I.E.

```
                READ(0,10.10)VIR
        OR      DIM VIR(0.0)
```

ARRAYS

THE ARRAY NAMING CONVENTION IS THE SAME AS FOR VARIABLES (AS DISCUSSED PREVIOUSLY). ALL LEGAL VARIABLE NAMES ARE ALSO LEGAL ARRAY NAMES.

ANALOG DEVICE MNEMONICS AND FUNCTIONS HAVE PRE-DEFINED MEANINGS WHICH ARE NO LONGER VALID IF THEIR NAMES ARE USED AS ARRAYS. THE VIRTUAL ARRAY FUNCTION "VIR" CAN BE DIMENSIONED AS AN ORDINARY ARRAY. HOWEVER, THEREAFTER IT CAN NO LONGER BE USED TO REFERENCE VIRTUAL ARRAY FILES.

IHI ALLOWS ONE AND TWO DIMENSIONED ARRAYS WITH THE MAXIMUM SUBSCRIPT VALUES DEFINED BY THE "DIM" STATEMENT (SEE CHAPTER 7). SUBSCRIPTS CAN RANGE FROM 0 TO THE MAXIMUM VALUES DEFINED IN THE "DIM" STATEMENT.

ARRAY ELEMENTS ARE REFERENCED AS FOLLOWS:

```
                DIM A(1,8),B(10)        ;SPECIFY MAXIMUM SUBSCRIPT VALUES
                A(1,7)=1        ;STORE A 1 IN ELEMENT (1,7) OF ARRAY "A"
                PRE B(I)        ;PRINT OUT ELEMENT "I" OF ARRAY "B"
                PRE A           ;PRINT OUT A(0,0)
```

INTERNALLY ALL VARIABLES ARE ACTUALLY TWO DIMENSIONED ARRAYS. HOWEVER, ORDINARY VARIABLES ARE 0 BY 0, AND ONE DIMENSION ARRAYS ARE OF DIMENSION N BY 0. FOR EXAMPLE:

```
                DIM A(0,0)
```

CREATES THE ORDINARY VARIABLE "A" AND

```
                DIM A(10,0)
```

IS EQUIVALENT TO:

```
                DIM A(10)
```

WHEN AN ARRAY IS USED IN AN EXPRESSION, ANY MISSING SUBSCRIPTS ARE ASSUMED TO BE 0, THAT IS:

```
        "A"    IS EQUIVALENT TO "A(0,0)"
   AND  "A(1)" IS EQUIVALENT TO "A(1,0)"
```

ARRAYS ARE STORED IN MEMORY "ROW" BY "ROW", WHERE THE FIRST ARRAY SUBSCRIPT IS THE "COLUMN" INDEX AND THE SECOND IS THE "ROW" INDEX:

```
        1-DIMENSIONAL ARRAY A(5) :

                A(0,0), A(1,0), A(2,0), A(3,0), A(4,0), A(5,0)

        2-DIMENSIONAL ARRAY A(1,1) :

                A(0,0), A(1,0), A(0,1), A(1,1)
```

ONCE AN ARRAY IS DIMENSIONED, IT CAN NOT BE RE-DIMENSIONED WITH DIFFERENT MAXIMUM SUBSCRIPT VALUES.

ARRAY SUBSCRIPTS CAN BE SPECIFIED AS CONSTANTS, VARIABLES, OR EXPRESSIONS. IF THE VALUE SPECIFIED IS NOT AN INTEGER, IT IS TRUNCATED TO AN INTEGER VALUE. AT RUN TIME ARRAY SUBSCRIPTS ARE CHECKED TO SEE THAT THEY ARE WITHIN THE MAXIMUM LIMITS SPECIFIED IN THE "DIM" STATEMENT. IF THEY ARE NOT, THEN AN ERROR MESSAGE IS PRINTED.

CHAPTER 4
MATHEMATICAL OPERATORS AND ARITHMETIC EXPRESSIONS

AN ARITHMETIC EXPRESSION CAN CONSIST OF A CONSTANT, A VARIABLE, A FUNCTION, OR A COMBINATION OF MATHEMATICAL OPERATORS AND THEIR OPERANDS. THE OPERANDS CAN BE NUMERIC OR STRING CONSTANTS, VARIABLES, FUNCTIONS, OR EXPRESSIONS ENCLOSED IN ()'S. THE MATHEMATICAL OPERATORS ARE DEFINED BELOW IN ORDER OF DECENDING PRIORITY OF EXECUTION:

OPERATOR	MEANING
** OR ↑ (UP ARROW)	EXPONENTIATION (A**B OR A↑B) (INVALID FOR: A <= 0 OR B*LN(A) > 88.0)
* AND /	MULTIPLICATION (A*B) AND DIVISION (A/B)
-	UNARY MINUS (-3)
+ AND -	ADDITION (A+B) AND SUBTRACTION (A-B)

(NOTE: THIS PRIORITY STRUCTURE IS MATHEMATICALLY EQUIVALENT TO THAT IN FORTRAN IV)

AN ARITHMETIC EXPRESSION MAY SIMPLY CONSIST OF ONE ELEMENT:

```
A
2.4
SQR(4)
'4CHA'
```

OR IT MAY CONSIST OF AN ALTERNATING SEQUENCE OF OPERANDS AND OPERATORS.

THE MATHEMATICAL OPERATORS WERE NOT INTENDED TO BE USED FOR STRING MANIPULATION. HOWEVER, STRING CONSTANTS CAN BE USED AS OPERANDS IN ARITHMETIC EXPRESSIONS AND FLOATING POINT ARITHMETIC OPERATIONS CAN BE PERFORMED ON STRING CONSTANTS, BUT THE RESULTING CHARACTER STRINGS ARE QUITE DIFFICULT (IF NOT IMPOSSIBLE) TO PREDICT (JUST AS IN FORTRAN).

WHEN TWO ADJACENT OPERATIONS IN AN EXPRESSION HAVE THE SAME PRIORITY (*,/ OR +,-) THE OPERATIONS ARE PERFORMED FROM LEFT TO RIGHT; OTHERWISE THE HIGHEST PRIORITY OPERATION IS PERFORMED FIRST. UNARY MINUS (FOR EXAMPLE, -A) IS EVALUATED AS IF THE OPERAND WERE PRECEDED BY 0 (I.E. 0-A). UNARY PLUS (I.E. +A) IS SYNTACTICALLY ALLOWED. HOWEVER, THE "+" IS SIMPLY IGNORED IN EVALUATING THE EXPRESSION.

AN ARITHMETIC EXPRESSION MAY NOT CONTAIN ADJACENT MATHEMATICAL OPERATORS:

```
A*+3
```

HOWEVER, THE FOLLOWING WOULD BE MEANINGFUL:

```
A*(+3)
```

TO ILLUSTRATE OPERATOR PRECEDENCE, THE FOLLOWING EXPRESSION:

```
-A**2+B/C*10
```

IS EVALUATED IN THE ORDER GIVEN WITHIN THE PARENTHESES BELOW:

```
(0 - (A**2)) + (B/C)*10
```

WITH THE EXPRESSIONS WITHIN NESTED PARENTHESES BEING EVALUATED FROM THE INNERMOST EXPRESSION OUTWARDS.

MATHEMATICAL FUNCTIONS (SUCH AS "SIN", "COS", ETC. ; SEE CHAPTER 5) CAN BE USED AS OPERANDS IN ARITHMETIC EXPRESSIONS. THE FUNCTION IS EVALUATED, AND THE SINGLE VALUE RETURNED IS USED IN THE EVALUATION OF THE ARITHMETIC EXPRESSION. FOR EXAMPLE, SIN(0) RETURNS THE VALUE 0.0 IN AN EXPRESSION. THE ARGUMENT OF THE FUNCTION MAY ITSELF BE AN ARITHMETIC EXPRESSION.

PARENTHESES CAN BE USED TO CHANGE THE ORDER OF EVALUATION IN AN EXPRESSION. FOR EXAMPLE:

```
A*B + C**2
```

IS EVALUATED AS:

```
(A*B) + (C**2)
```

IF INSTEAD THE DESIRED OPERATION IS TO SQUARE "B" PLUS "C", AND TO MULTIPLY THE RESULT BY "A", PARENTHESES CAN BE USED AS SHOWN BELOW TO ACHIEVE THE DESIRED OPERATION:

```
A*(B+C)**2
```

PARENTHESES CANNOT BE USED TO IMPLY MULTIPLICATION, AS THEY ARE COMMONLY USED IN MATHEMATICS. FOR EXAMPLE:

```
SQR(9)
```

IS THE SQUARE ROOT OF 9, WHEREAS

```
SQR*(9)
```

INDICATES THE VALUE OF THE VARIABLE "SQR" MULTIPLIED BY 9.

IF AN OVERFLOW OCCURS WHEN AN OPERATION IN AN ARITHMETIC EXPRESSION IS BEING EVALUATED, THEN A WARNING MESSAGE IS PRINTED. IN THE CASE OF OVERFLOW (RESULT > APPROX. 1.7E+38) THE RESULT OF THE OPERATION IS ASSUMED TO BE THE LARGEST SINGLE PRECISION VALUE (ROUGHLY .1701E+39) AND THE COMPUTATION PROCEEDS USING THAT VALUE. IN THE CASE OF UNDERFLOW (RESULT < APPROX. .28E-38) THE RESULT IS ASSUMED TO BE 0.0 AND THE COMPUTATION PROCEEDS. AN ATTEMPT TO DIVIDE BY ZERO RESULTS IN A FATAL ERROR (I.E. THE STATEMENT BEING EXECUTED IS ABORTED).

CHAPTER 5
MATHEMATICAL LIBRARY FUNCTIONS

IHI PROVIDES MATHEMATICAL LIBRARY FUNCTIONS FOR USE IN ARITHMETIC EXPRESSIONS (MUCH THE SAME AS IN FORTRAN IV). THIS CHAPTER DEFINES THESE FUNCTIONS AND THEIR ARGUMENTS. EACH FUNCTION HAS ONE OR MORE ARGUMENTS WHICH CAN BE SPECIFIED AS CONSTANTS, VARIABLES, OR ARITHMETIC EXPRESSIONS. THE ARGUMENT OF A FUNCTION CAN ALSO BE ANOTHER FUNCTION OR AN ANALOG DEVICE MNEMONIC.

FUNCTION CALL		DEFINITION
ABS(X)		ABSOLUTE VALUE !X!
LN(X)	*	NATURAL LOGARITHM (BASE E) (WHERE X > 0.0)
LOG(X)		BASE 10 LOGARITHM (WHERE X > 0.0)
ATN(X)		ARCTANGENT OF X (RESULT IS ANGLE IN RADIANS)
EXP(X)	*	EXPONENTIAL (E**X) (WHERE X <= 88.0)
SIN(X)		SINE FUNCTION (X IS ANGLE IN RADIANS)
COS(X)		COSINE FUNCTION (X IS ANGLE IN RADIANS)
SQR(X)		SQUARE ROOT (WHERE X > 0.0)
INT(X)		INTEGER PART OF X (INT(X)=X-FRC(X))
FRC(X)		FRACTIONAL PART OF X (FRC(X)=X-INT(X))
SGN(X)		RETURNS +1.0 IF X > 0.0 RETURNS 0.0 IF X = 0.0 (DIFFERS FROM FORTRAN) RETURNS -1.0 IF X < 0.0
MAX(<LIST>)		EVALUATES ALL THE ARGUMENTS IN THE <LIST> AND RETURNS THE MAXIMUM VALUE. ANY NUMBER OF ARGUMENTS GREATER THAN OR EQUAL TO 2 IS ALLOWED.
MIN(<LIST>)		EVALUATES ALL THE ARGUMENTS IN THE <LIST> AND RETURNS THE MINIMUM VALUE. ANY NUMBER OF ARGUMENTS GREATER THAN OR EQUAL TO 2 IS ALLOWED.
TIM(X)		RETURNS THE NUMBER OF SECONDS AS RECORDED ON THE SYSTEM CLOCK SINCE X SECONDS PAST MIDNIGHT. (TIME SINCE MIDNIGHT = TIM(0))
VIR(X)		READS OR WRITES RECORD X IN THE CURRENT VIRTUAL ARRAY FILE (SEE "VIRTUAL ARRAYS" IN CHAPTER 7).

*NOTE: E IS THAT FAMOUS CONSTANT 2.718281828459045 (ROUGHLY).

THE FOLLOWING ARE SOME EXAMPLE USES OF MOST OF THESE FUNCTIONS:

```
        10.10 START=TIM(0)               ;START=SECONDS PAST MIDNIGHT
        10.20 E=EXP(1)                   ;GET THE VALUE OF "E"
        10.30 PRI LN(EXP(1))             ;SHOULD PRINT 1
        10.40 TAN=SIN(X)/COS(X)          ;COMPUTE TANGENT OF "X"
        10.45 PIE=4*ATN(1)               ;COMPUTES THE VALUE OF "PIE"
        10.50 I=SQR(-1)                  ;THIS IS A NO-NO !!!
        10.60 ONE=ABS(-1)                ;AN UNUSUAL WAY TO COMPUTE 1
        10.70 TWO=2*LOG(10)              ;A HARD WAY TO COMPUTE 2
        10.80 PRI SGN(-1),SGN(0),SGN(1) ;SEE HOW "SGN" WORKS
        10.85 VIR(1)=-VIR(1)             ;INVERTS VALUE IN RECORD 1
        10.90 X=INT(X)+FRC(X)            ;SHOULD NOT MODIFY "X"
        10.92 BIG=MAX(1,2,3)             ;SET BIG = 3
        10.94 SMALL=MIN(1,2,3)           ;SET SMALL = 1
        10.96 END=TIM(START)             ;END = SECONDS TO EXECUTE THIS
```

CHAPTER 6
HYBRID FUNCTIONS

THIS CHAPTER DESCRIBES THE SYNTAX AND USE OF THE ANALOG DEVICE MNEMONICS AND HOW THE ARE HANDLED BY IHI. FOR A DETAILED DESCRIPTION OF THE SPECIFIC ANALOG DEVICE MNEMONICS IMPLEMENTED IN IHI REFER TO APPENDIX B.

AN ANALOG DEVICE MNEMONIC CONSISTS OF A THREE LETTER MNEMONIC CODE FOLLOWED BY EXACTLY THREE NUMBERS AND AN OPTIONAL ANALOG CONSOLE NUMBER IN ()'S AS FOLLOWS:

```
                DEVNNN
        OR      DEVNNN(<EXPRESSION>)
```

THE THREE DIGIT DECIMAL NUMBER MUST BE A POTENTIALLY VALID ANALOG ADDRESS, OR THE SYMBOL IS AN ORDINARY VARIABLE BY DEFAULT. THUS THE FOLLOWING ARE ANALOG DEVICE MNEMONICS:

```
                COF000
                POT000(0)
```

AND THE FOLLOWING ARE NOT ANALOG DEVICE MNEMONICS:

```
                COF0
                COF0(0)
                POT999
```

IF NO ANALOG CONSOLE NUMBER IS SPECIFIED, THEN THE CURRENT ANALOG CONSOLE NUMBER IS ASSUMED. THAT IS THE CONSOLE NUMBER "I" IN THE LAST "CALL CONSO(I)" OR THE SMALLEST ANALOG CONSOLE NUMBER ATTACHED IF NO CALLS TO "CONSO" HAVE TAKEN PLACE. THUS,

```
                COF000
```

REFERS TO COEFFICIENT 000 ON THE CURRENT ANALOG CONSOLE AND

```
                POT001(J)
```

IS POT 001 ON CONSOLE "J".

THERE ARE TWO GROUPS OF ANALOG DEVICE MNEMONICS; THOSE THAT CAN BE SET AND THOSE THAT CAN ONLY BE READ.

THE "READABLE" ANALOG DEVICE MNEMONICS CAN ONLY APPEAR IN AN EXPRESSION IN A "READ" CONTEXT. IF ONE IS USED TO THE LEFT OF AN "=" IN AN ASSIGNMENT STATEMENT AS FOLLOWS:

```
                ADC000=ADC001
```

THEN IN THIS CASE THE VARIABLE "ADC000" IS CREATED, AND "ADC000" CAN NO LONGER BE USED TO READ ADC CHANNEL 000.

THE FOLLOWING EXAMPLES ILLUSTRATE SOME USES OF THE SETTABLE ANALOG DEVICE MNEMONICS:

```
        COF000 = ADC000  ;READ ADC000 AND SET COEF 000 TO THE VALUE
        COF000(1) = 1.2340
        WHEN (ADC000'LT'0) COF000=-COF000  ;INVERT COEFF SETTING WHEN
                                           ;ADC READING GOES NEGATIVE
```

SETTABLE ANALOG DEVICE MNEMONICS DO NOT CREATE A VARIABLE
IF THEY ARE BEING USED TO SET A DEVICE.

CHAPTER 7
IHI LANGUAGE STATEMENTS

THIS CHAPTER DEFINES THE IHI LANGUAGE STATEMENTS. MOST STATEMENTS CAN BE USED IN BOTH THE "IMMEDIATE" AND "DEFERRED" MODES OF OPERATION. HOWEVER, SOME STATEMENTS MUST LOGICALLY BE RESTRICTED TO "DEFERRED" MODE PROGRAMS, SUCH AS "FOR - NEXT LOOPS", SUBROUTINES, ETC... AND ANY SUCH RESTRICTIONS WILL BE INDICATED IN THE STATEMENT DESCRIPTION.

NO-OP STATEMENT

```
<BLANKS OR TABS>;
;
NOP
```

THE NO-OP STATEMENT CAUSES NO OPERATION AT RUN TIME. THE FOLLOWING DEFERRED MODE PROGRAM ILLUSTRATES THE USE OF SEVERAL FORMS OF THE NO-OP STATEMENT:

```
10.00 ;THIS PROGRAM DOES NOTHING
10.10 ;
10.20 NOP       ; THIS STATEMENT DOES NOTHING
10.25 ;
10.30 ;A=1      ;STATEMENTS CAN BE NO-OP'ED LIKE THIS
10.35 ;
10.40 ;NOP      ;NOP'S CAN BE USED TO RESERVE STORAGE
10.50 ;NOP      ;FOR STATEMENTS.  THESE NOP'S CAN LATER
10.60 ;NOP      ;BE OVERLAYED WITH MEANINGFUL STATEMENTS.
10.80 STOP
```

DIMENSION STATEMENT

```
DIM  VAR1(I1[,J1]),VAR2(I2[,J2]),....,VARN(IN[,JN])
```

WHERE I AND J INDICATE THE MAXIMUM SUBSCRIPT VALUES FOR EACH ARRAY (",J" IS OPTIONAL AND DEFAULTS TO 0).

THE DIMENSION STATEMENT ALLOCATES STORAGE SPACE FOR ARRAY VARIABLES. IHI SUPPORTS ONE AND TWO-DIMENSIONED ARRAYS AND SUBSCRIPTS CAN RANGE FROM 0 TO THE LIMIT DEFINED IN THE DIMENSION STATEMENT. THE MAXIMUM DIMENSIONS MUST BE IN THE RANGE 0 TO 255 INCLUSIVE, OR ELSE AN ERROR MESSAGE IS PRINTED.

A SINGLE DIMENSION ARRAY "A" CONSISTING OF 10 STORAGE LOCATIONS REFERENCED AS A(0) TO A(9) IS DIMENSIONED AS FOLLOWS:

```
DIM A(9)
```

A 10 BY 10 MATRIX "B" (100 REAL STORAGE LOCATIONS) IS DIMENSIONED AS FOLLOWS:

```
DIM B(9,9)
```

THE FOLLOWING EXAMPLE CREATES AN ORDINARY VARIABLE "C" (ONE STORAGE LOCATION):

```
DIM C(0,0)
```

THE SUBSCRIPT VALUES CAN ALSO BE SPECIFIED AS VARIABLES OR ARITHMETIC EXPRESSIONS AS ILLUSTRATED IN THE FOLLOWING EXAMPLE:

```
ONE=1
TWO=2
DIM A(ONE,TWO),B(2*TWO,3+2*6)
```

ONCE A VARIABLE HAS BEEN DIMENSIONED VIA THE "DIM" STATEMENT (OR ONCE AN ORDINARY VARIABLE HAS BEEN CREATED) IT CANNOT BE RE-DIMENSIONED WITH DIFFERENT MAXIMUM SUBSCRIPT VALUES.

IT SHOULD BE MENTIONED THAT INTERNALLY ALL VARIABLES ARE TWO-DIMENSIONED, BUT SOME ARE 0 BY 0 (SUCH AS ORDINARY VARIABLES) AND SOME ARE N BY 0 (SUCH AS ONE-DIMENSIONED ARRAYS). ALSO NOTE THAT "VAR","VAR(0)", AND "VAR(0,0)" ALL REFERENCE THE SAME STORAGE LOCATION, NAMELY THE FIRST STORAGE LOCATION IN THE ARRAY "VAR".

WHEN A SUBSCRIPTED VARIABLE IS USED IN A PROGRAM, THE SUBSCRIPTS ARE CHECKED TO SEE THAT THEY ARE WITHIN THE MAXIMUM LIMITS SPECIFIED IN THE DIMENSION STATEMENT. IF THEY ARE NOT, AN ERROR MESSAGE IS PRINTED.

ASSIGNMENT STATEMENT

```
    <VARIABLE>=<EXPRESSION>
LET <VARIABLE>=<EXPRESSION>
```

THE ASSIGNMENT STATEMENT ASSIGNS A VALUE TO A VARIABLE OR SETS AN ANALOG COEFFICIENT DEVICE. IN THE CASE OF ORDINARY VARIABLES, IF THE VARIABLE DOES NOT EXIST PRIOR TO THE ASSIGNMENT STATEMENT, THEN A SYMBOL TABLE ENTRY IS CREATED. ARRAY VARIABLES MUST BE DIMENSIONED PRIOR TO THFIR USE AS ARRAYS. IF THE VARIABLE IS A SETTABLE ANALOG DEVICE MNEMONIC BEING USED TO SET AN ANALOG COEFFICIENT DEVICE, THEN NO SYMBOL TABLE ENTRY IS CREATED (SEE CHAPTER 6).

SOME EXAMPLES OF ASSIGNMENT STATEMENTS ARE:

```
A=2.0
LET LET=A**2
B(A,LET)=A+1
COF000=1.0       ;SET COEFF 000 TO 1.0
COF001(1)=.1     ;SET COEFF 001 ON CONSOLE 1 TO 1.0
C(A)=B(A,LET)
ASCII(1)='THIS'          ;STORE THE FOLLOWING STRING OF
ASCII(2)=' IS '          ;CHARACTERS IN THE ARRAY "ASCII":
ASCII(3)='AN A'          ;"THIS IS AN ASCII STRING."
ASCII(4)='SCII'
ASCII(5)=' STR'
ASCII(6)='ING.'
```

JUMP STATEMENT

```
JUMP GG.LL
JUMP GG
```

THIS STATEMENT CAUSES THE TRANSFER OF CONTROL TO LINE "GG.LL" (FIRST SYNTAX FORM) OR TO THE FIRST STATEMENT IN GROUP "GG" (SECOND SYNTAX FORM). NOTE THAT A "JUMP GG" AND "JUMP GG.00" ARE EQUIVALENT.

A JUMP TO THE SAME LINE AS THE "JUMP" STATEMENT IS NOT ALLOWED. I.E.

```
10.10 IF (AMP000'GT'0) JUMP 10.10
```

HOWEVER, THIS RESTRICTION MAY BE EASILY CIRCUMVENTED BY PRECEDING A "JUMP" STATEMENT WITH A NO-OP. I.E.

```
10.09 NOP
10.10 IF (AMP000'GT'0) JUMP 10.09
```

THE "JUMP" STATEMENT CAN BE USED IN THE "IMMEDIATE" MODE TO RESTART A DEFERRED MODE PROGRAM AT ANY STATEMENT NUMBER FOLLOWING AN ERROR MESSAGE OR THE EXECUTION OF A "STOP" OR "PAUSE" STATEMENT. THIS IS A USEFUL DEBUGGING FEATURE; HOWEVER, CARE MUST BE TAKEN IN INTERPRETING THE RESULTS OF SUCH AN OPERATION.

ANOTHER USEFUL FEATURE OF THE "JUMP" STATEMENT IS THAT THE STATEMENT NUMBER CAN BE SPECIFIED AS A VARIABLE OR ARITHMETIC EXPRESSION. THIS ALLOWS THE USER TO COMPUTE THE STATEMENT NUMBER FOLLOWING THE "JUMP" AT RUN TIME, FOR EXAMPLE, THE FOLLOWING ARE BOTH LEGAL:

```
            JUMP I
    OR      JUMP GROUP+LINE/100.
```

CONDITIONAL STATEMENT ("DEFERRED" MODE ONLY)

WHEN(<EXPRESSION><RELATION><EXPRESSION>) <STATEMENT>

WHERE <STATEMENT> IS ANY STATEMENT CAPABLE OF BEING EXECUTED IN THE "DEFERRED" MODE (EXCEPT FOR A DEFAULT NO-OP, WHERE THE FIRST NON-BLANK CHARACTER FOLLOWING THE "WHEN ( )" IS ";").

THE "WHEN" STATEMENT ALLOWS CONDITIONAL EXECUTION OF THE FOLLOWING <STATEMENT> DEPENDING ON WHETHER OR NOT THE TESTED <RELATION> BETWEEN THE VALUES OF THE <EXPRESSION>S IS SATISIFIED.

THE FOLLOWING TABLE DEFINES THE <RELATION>S THAT CAN BE TESTED:

RELATIONAL OPERATOR	EXAMPLE	MEANING
'EQ'	A 'EQ' B	TESTS IF "A" IS EQUAL TO "B" (A = B)
'NE'	A 'NE' B	TESTS IF "A" IS NOT EQUAL TO "B" (A <> B)
'LT'	A 'LT' B	TESTS IF "A" IS LESS THAN "B" (A < B)
'LE'	A 'LE' B	TESTS IF "A" IS LESS THAN OR EQUAL TO "B" (A < B OR A = B)
'GT'	A 'GT' B	TESTS IF "A" IS GREATER THAN "B" (A > B)
'GE'	A 'GE' B	TESTS IF "A" IS GREATER THAN OR EQUAL TO "B" (A > B OR A = B)
'AE'	A 'AE' B	TESTS IF "A" IS ALMOST EQUAL TO "B" (ABS(A-B) > 0.0001).
'NA'	A 'NA' B	TESTS IF "A" IS NOT ALMOST EQUAL TO "B" (INVERSE OF 'AE')

THE FOLLOWING EXAMPLE ILLUSTRATES HOW THE LOGICAL "AND" OR "OR" OF SEVERAL RELATIONS CAN BE IMPLEMENTED:

```
10.10 ; NESTED "WHEN" STATEMENTS YIELD A LOGICAL "AND"
10.20 ; OF THE RELATIONS TESTED.
10.25 ;
10.30 WHEN (A+9'GT'27) WHEN (SIN(ALPHA)'LT'0) ALPHA=-ALPHA
10.40 ;
10.50 ; SEQUENTIAL "WHEN" STATEMENTS EACH FOLLOWED BY A
10.55 ; "JUMP" TO THE SAME STATEMENT NUMBER, YIELD A LOGICAL
10.57 ; "OR" OF THE RELATIONS TESTED.
10.65 ;
10.70 WHEN (N1'GT'MAXNUM) JUMP 11.01     ;JUMP TO 11.01 IF EITHER
10.80 WHEN (N2'GT'MAXNUM) JUMP 11.01     ;N1 OR N2 OR N3 IS
10.90 WHEN (N3'GT'MAXNUM) JUMP 11.01     ;GREATER THAN MAXNUM
10.95 GOSUB DOIT  ;AND DO IT AGAIN
10.97 JUMP 10.70
11.01 GOSUB DIDIT ;NOW GO CHECK SOME OTHER THINGS
11.02 WHEN (ITIS'EQ'ENOUGH) STOP
11.03 JUMP 10.95
```

FOR-NEXT LOOPS ("DEFERRED" MODE ONLY)

```
FOR <VARIABLE>=<EXP1>,<EXP2>[,<EXP3>]
NEXT <VARIABLE>
```

WHERE <VARIABLE> IS THE LOOP INDEX VARIABLE, <EXP1> IS ITS STARTING VALUE, <EXP2> IS ITS MAXIMUM VALUE, AND OPTIONAL <EXP3> IS THE VALUE ADDED TO <EXP1> EACH ITERATION (<EXP3> DEFAULTS TO 1).

THE COMBINATION OF A "FOR" STATEMENT AND A TERMINATING "NEXT" STATEMENT CONSTITUTES AN ITERATIVE COMPUTING LOOP. EACH TIME THE "NEXT" STATEMENT IS EXECUTED, THE CORRESPONDING LOOP INDEX VARIABLE IS INCREMENTED BY THE VALUE OF <EXP3>. WHEN THE RESULT IS OUTSIDE EITHER THE RANGE <EXP1> TO <EXP2> OR <EXP2> TO <EXP1>, THE STATEMENT FOLLOWING THE "NEXT" IS EXECUTED; OTHERWISE, CONTROL LOOPS BACK TO THE STATEMENT FOLLOWING THE "FOR".

NOTE THAT THERE IS NO RESTRICTION ON THE SIGN OF THE INDEX <VARIABLE> OR THE VALUES OF THE <EXP>'S IN THE "FOR" STATEMENT. FOR EXAMPLE, THE FOLLOWING IS VALID:

```
FOR I=0,-10.2,-.2
```

THE <EXP>'S IN THE "FOR" STATEMENT ARE ONLY EVALUATED ONCE: THE FIRST TIME THE "FOR" STATEMENT IS EXECUTED; THEY ARE NOT EVALUATED DYNAMICALLY EACH ITERATION THROUGH THE LOOP.

UPON EXIT FROM THE LOOP VIA THE NORMAL LOOP TERMINATING CONDITION, THE INDEX <VARIABLE> HAS BEEN INCREMENTED BEYOND THE VALUE OF <EXP2> TO A FINAL VALUE OF: <EXP2> + <EXP3>.

NESTED LOOPS ARE ALLOWED, FOR EXAMPLE:

```
!-----FOR I=1,10
!
!  !--FOR J=1,10
!  !
!  !--NEXT J
!
!-----NEXT I
```

HOWEVER, LOOPS CAN NOT OVERLAP ONE ANOTHER AS THE FOLLOWING ILLUSTRATES:

```
!-----FOR I=1,10
!
! !---FOR J=1,10
! !
!-!---NEXT I
  !
  !---NEXT J
```

THE FOLLOWING EXAMPLE ILLUSTRATES THE USE OF "FOR-NEXT" LOOPS TO INTERCHANGE THE ROWS AND COLUMNS OF A MATRIX:

```
10.10   DIM M1(1,1),M2(1,1)
10.11   M1(0,0)=1
10.12   M1(0,1)=2
10.13   M1(1,0)=3
10.14   M1(1,1)=4
10.50   WRITE(1,10.60)M1
10.60   FORMAT(' ORIGINAL MATRIX:'/(2I10/))

20.10   FOR I=0,1
20.20   FOR J=0,1
20.30   M2(I,J)=M1(J,I)
20.40   NEXT J
20.50   NEXT I

30.10   WRITE(1,30.20)M2
30.20   FORMAT(' ROWS AND COLUMNS INTERCHANGED:'/(2I10/))
30.30   STOP

?RUN
ORIGINAL MATRIX:
        1         3
        2         4
ROWS AND COLUMNS INTERCHANGED:
        1         2
        3         4

IHI --WARN-- STOP AT LINE 30.30
?
```

PAUSE STATEMENT ("DEFERRED" MODE ONLY)

```
PAUSE
```

THE "PAUSE" STATEMENT CAUSES THE MESSAGE :

```
IHI --WARN-- PAUSE AT GG.LL
```

AND CONTROL IS RETURNED TO "IMMEDIATE" MODE FOR A COMMAND FROM THE CONSOLE OPERATOR (JUST AS IF A "STOP" STATEMENT HAD BEEN EXECUTED). TO CONTINUE EXECUTION OF THE "DEFERRED" MODE PROGRAM AT THE STATEMENT FOLLOWING THE "PAUSE", AN "IMMEDIATE" MODE CONTINUE "CON" COMMAND CAN BE TYPED. SEE CHAPTER 2 FOR A DESCRIPTION OF THE "IMMEDIATE" MODE "CON" COMMAND.

THE "PAUSE" STATEMENT IS USEFUL FOR DEBUGGING PURPOSES SINCE IT ALLOWS THE USER TO EXAMINE THE VALUES OF VARIABLES, CHANGE THE VALUES OF SOME VARIABLES IF NECESSARY, OR EVEN DELETE OR ENTER ADDITIONAL EXECUTABLE "DEFERRED" MODE STATEMENTS, AND THEN CONTINUE THE EXECUTION OF THE PROGRAM. THE PROGRAM CAN BE CONTINUED USING AN IMMEDIATE MODE "JUMP" OR "GOSUB" STATEMENT. HOWEVER, CARE MUST BE TAKEN IN PROPERLY INTERPRETING THE RESULTS OF SUCH AN OPERATION.

THE FOLLOWING EXAMPLE DEFERRED MODE PROGRAM PAUSES AND EXPECTS THE OPERATOR TO EXECUTE AN "IMMEDIATE" MODE ASSIGNMENT STATEMENT DEFINING THE VALUE OF THE VARIABLE "A". THE "IMMEDIATE" MODE 'CON' STATEMENT CONTINUES THE PROGRAM AT LINE 93.10.

```
91.10 WRITE(1,91.20)
91.20 FORMAT(' ASSIGN A VALUE TO A'/)
92.10 PAUSE
93.10 SINE=SIN(A)
93.20 COSINE=COS(A)
93.30 PRI SINE,COSINE
93.40 JUMP 91.10

?RUN
ASSIGN A VALUE TO A

IHI --WARN-- PAUSE AT LINE 92.10
A=0
CON
SINE =        0
COSINE =        1
ASSIGN A VALUE TO A

IHI --WARN-- PAUSE AT LINE 92.10
```

STOP STATEMENT ("DEFERRED" MODE ONLY)

```
STOP
```

THE "STOP" STATEMENT HALTS THE "DEFERRED" MODE PROGRAM, CAUSES THE MESSAGE (WHERE GG.LL IS THE LINE NUMBER OF THE "STOP" STATEMENT):

```
IHI --WARN-- STOP AT LINE GG.LL
```

AND RETURNS TO THE "IMMEDIATE" MODE FOR A COMMAND FROM THE OPERATOR. THE "STOP" STATEMENT DOES NOT CLEAR OUT THE SYMBOL TABLE. THIS ALLOWS THE USER TO EXAMINE THE RESULTS OF THE EXECUTION OF THE "DEFERRED" MODE PROGRAM FROM THE "IMMEDIATE" MODE. THE "STOP" STATEMENT IS NOT REQUIRED TO TERMINATE A "DEFERRED" MODE PROGRAM, HOWEVER WHEN DEBUGGING COMPLEX LOOPING ALGORITHMS, A CONDITIONAL "STOP" OR "PAUSE" STATEMENT IS HIGHLY RECOMMENDED SOMEWHERE IN SUCH A LOOP.

THE FOLLOWING PROGRAM ILLUSTRATES THE USE OF THE "STOP" STATEMENT. NOTE THAT THE STATEMENT NUMBER PRINTED WILL IDENTIFY WHICH "STOP" WAS EXECUTED:

```
30.10 WHEN (ER10.GT.0) STOP
30.20 WHEN (ER20.GT.0) STOP
30.30 WHEN (ER30.GT.0) STOP
40.10 STOP
```

IF A "DEFERRED" MODE PROGRAM COMES TO THE LAST STATEMENT, AND IT IS NOT A "STOP" STATEMENT, THEN THE EQUIVALENT OF A "STOP" STATEMENT IS EXECUTED AND THE STOP MESSAGE IS PRINTED:

```
IHI --WARN-- STOP AT LINE GG.LL
```

WHERE GG.LL IS THE LAST STATEMENT EXECUTED.

CALL STATEMENT

CALL <NAME>(<ARGUMENT LIST>)

THE "CALL" STATEMENT CALLS THE PRE-DEFINED SYSTEM SUBROUTINE <NAME>. SEE APPENDIX C FOR A DESCRIPTION OF THE SUBROUTINES WHICH HAVE BEEN IMPLEMENTED IN IHI. THE FORM OF THE ARGUMENT LIST IS PRE-DEFINED DEPENDING ON THE SUBROUTINE, AND MAY INCLUDE CONSTANTS, VARIABLES, OR EXPRESSIONS. THE CALL SEQUENCE IS THE SAME AS IN FORTRAN IV. IT IS POSSIBLE FOR THE USER TO IMPLEMENT HIS OWN SUBROUTINES WRITTEN IN MACRO ASSEMBLY LANGUAGE OR FORTRAN IV; HOWEVER, THIS REQUIRES EDITING AND RE-TASK BUILDING THE IHI LANGUAGE PROCESSOR. THE INFORMATION NECESSARY TO MAKE SUCH A CHANGE TO THE IHI LANGUAGE PROCESSOR WILL BE SUPPLIED WITH THE SOURCE LISTINGS OF THE IHI LANGUAGE PROCESSOR.

IHI SUBROUTINES

```
GOSUB <NAME>

SUBR <NAME>     ("DEFERRED" MODE ONLY)
RETURN          ("DEFERRED" MODE ONLY)
```

THE "GOSUB" STATEMENT TRANSFERS CONTROL TO THE DEFERRED MODE IHI SUBROUTINE <NAME>. NO ARGUMENT LIST IS NECESSARY SINCE ALL VARIABLES IN THE MAIN PROGRAM (OR CURRENT SYMBOL TABLE) ARE "GLOBAL" TO ALL IHI SUBROUTINES.

THE "SUBR" STATEMENT ("DEFERRED" MODE ONLY) INDICATES THE START OF AN IHI SUBROUTINE AND ITS <NAME>. THE "RETURN" STATEMENT ("DEFERRED" MODE ONLY) INDICATES THE END OF A SUBROUTINE. WHEN SUBROUTINE <NAME> EXECUTES A "RETURN" STATEMENT, CONTROL IS RETURNED TO THE STATEMENT IMMEDIATELY FOLLOWING THE "GOSUB".

IF THE "GOSUB" IS EXECUTED IN THE "IMMEDIATE" MODE, CONTROL IS RETURNED TO THE "IMMEDIATE" MODE FOLLOWING EXECUTION OF THE "RETURN" STATEMENT. AN "IMMEDIATE" MODE "GOSUB" IS A CONVENIENT MEANS OF CHECKING OUT AN IHI SUBROUTINE, PRIOR TO CALLING IT FROM A "DEFERRED" MODE PROGRAM. CARE MUST BE TAKEN TO SET UP ALL PARAMETERS USED BY THE SUBROUTINE BEFORE EXECUTING THE "IMMEDIATE" MODE "GOSUB".

THE FOLLOWING EXAMPLE ILLUSTRATES A SUBROUTINE CALL:

```
10.10 A=10
10.20 GOSUB FACTAL       ;COMPUTE 10!
10.25 PRI AFACT          ;PRINT OUT RESULT
10.30 STOP

20.10 SUBR FACTAL        ;COMPUTE "A" FACTORIAL (A!)
20.20 ALESS1=A
20.30 AFACT=A
20.40 ALESS1=ALESS1-1
20.50 WHEN (ALESS1'LE'0) RETURN
20.60 AFACT=AFACT*ALESS1
20.70 JUMP 20.40

?RUN
AFACT = 3628800
IHI --WARN-- STOP AT LINE 10.30
?A = 3
?GOSUB FACTAL   ;COMPUTE 3!
?PRI AFACT      ;PRINT RESULT
AFACT =       6
?
```

INPUT/OUTPUT OPERATIONS

FORMAT STATEMENT ("DEFERRED" MODE ONLY)

```
FORMAT(<LIST>)
FORMAT(<LIST1>(<LIST2>))
```

WHERE THE <LIST>S CONSIST OF FORMAT SPECIFIERS "S"
SEPARATED BY "," <BLANK> OR "/" AS FOLLOWS:
S,S S/S/....S/S/S,S,S/

THIS STATEMENT DESCRIBES HOW THE FORMATTED ASCII RECORDS ARE TO BE USED WITH THE ASSOCIATED "READ" OR "WRITE" STATEMENT. IN THE FIRST SYNTAX FORM, IF THE <LIST> IS EXHAUSTED, IT IS RE-USED STARTING FROM THE BEGINNING. IN THE SECOND SYNTAX FORM, ONLY THE PARENTHESIZED <LIST2> IS RE-USED.

FOR USERS FAMILIAR WITH FORTRAN IV, THE FORMAT <LIST> SPECIFICATIONS ARE VERY SIMILIAR TO THOSE USED IN FORTRAN FORMATS. FOR USERS NOT FAMILIAR WITH FORTRAN, THE REST OF THIS SECTION SHOULD CLARIFY THE FORM OF THE FORMAT <LIST>S.

DELIMITERS

THE FOLLOWING DELIMITERS ARE ALLOWED BETWEEN SPECIFIERS IN A FORMAT <LIST>:

","	COMMAS DO NOTHING BUT SEPARATE LIST ITEMS
" "	BLANKS WILL SEPARATE LIST ITEMS AND ARE IGNORED BETWEEN ITEMS ONLY.
"/"	A SLASH IS USED TO CAUSE THE NEXT RECORD TO BE READ FOR "READ" STATEMENT FORMATS OR THE CURRENT RECORD TO BE OUTPUT FOR "WRITE" STATEMENTS.

SPECIFIERS FOR "READ" OR "WRITE" FORMATS

THE FOLLOWING FORMAT <LIST> SPECIFIERS ARE AVAILABLE FOR EITHER "READ" OR "WRITE" STATEMENT FORMATS:

"A" FORMAT SPECIFIER

```
[REPEAT COUNT]A<WIDTH>
```

THE "A" SPECIFIER CAUSES <WIDTH> CHARACTERS (1-4) TO BE TRANSFERRED WITH NO CONVERSION. THE [REPEAT COUNT] IS OPTIONAL AND DEFAULTS TO 1 (I.E. "A4" IS EQUIVALENT TO "1A4").

THE "A" SPECIFIER IS USED TO READ OR WRITE ASCII RECORDS. FROM 1 TO 4 ASCII CHARACTERS CAN BE STORED LEFT JUSTIFIED IN A REAL VARIABLE.

"X" FORMAT SPECIFIER

```
[REPEAT COUNT]X
```

THE "X" SPECIFIER CAUSES [REPEAT COUNT] BLANKS TO BE INSERTED IN AN OUTPUT LINE OR THE NEXT [REPEAT COUNT] CHARACTERS TO BE IGNORED IN AN INPUT LINE. THE [REPEAT COUNT] IS OPTIONAL AND DEFAULTS TO 1.

SPECIFIERS FOR "WRITE" STATEMENT FORMATS

CHARACTER STRINGS

ASCII CHARACTER STRINGS, DELIMITED BY SINGLE QUOTES (') ONLY, ARE ALLOWED IN "WRITE" FORMATS MUCH LIKE IN FORTRAN IV. A SINGLE QUOTE CAN BE INCLUDED IN THE STRING BY USING TWO ADJACENT QUOTES TO INDICATE THAT ONE QUOTE IS TO BE RETAINED IN THE STRING:

```
                'THIS IS A STRING'       IS THE STRING: THIS IS A STRING
                'DON''T'                 IS THE STRING: DON'T
        AND     ''''                     IS THE STRING: '
```

CARRIAGE CONTROL

IN "WRITE" STATEMENT FORMATS THE FIRST ITEM IN THE FORMAT LIST IS A CARRIAGE CONTROL CHARACTER (JUST AS IN FORTRAN IV) DEFINED AS FOLLOWS:

CHARACTER	EFFECT
<BLANK>	CARRIAGE RETURN, LINE FEED (ADVANCE PRINTING POSITION TO THE BEGINNING OF THE NEXT LINE)
'0'	CARRIAGE RETURN, 2 LINE FEEDS
'1'	CARRIAGE RETURN, TOP OF FORM (ADVANCE PRINTING POSITION TO THE BEGINNING OF THE NEXT PAGE)
'+'	CARRIAGE RETURN (ADVANCE PRINTING POSITION TO THE BEGINNING OF THE CURRENT LINE; THIS ALLOWS THE LINE TO BE OVERPRINTED)
'$'	INHIBIT CARRIAGE CONTROL (USED FOR EXAMPLE WHEN PRINTING LINES THAT INTERACT WITH THE USER AT THE CONSOLE)

NOTE : IF THE CARRIAGE CONTROL CHARACTER IS NOT EXPLICITLY SPECIFIED AS THE FIRST CHARACTER IN THE "WRITE" STATEMENT FORMAT, THEN THE FIRST ASCII CHARACTER IN THE CONVERTED OUTPUT LINE WILL BE USED AS THE CARRIAGE CONTROL CHARACTER AND WILL NOT BE PRINTED.

"E", "F", "H", AND "I" SPECIFIERS

THESE SPECIFIERS ARE ONLY FOR USE IN "WRITE" STATEMENT FORMATS. THEY CAUSE <WIDTH> CHARACTERS TO BE OUTPUT RIGHT JUSTIFIED IN THE FIELD. IF <WIDTH> IS NOT ENOUGH ROOM FOR THE VALUE TO BE PRINTED, THE FIELD IS FILLED WITH "*"S. NOTE THAT THE LEAST SIGNIFICANT DIGIT PRINTED IS ROUNDED UP IF THE NEXT NON-PRINTED DIGIT IS >= 5.

"E" FORMAT SPECIFIER

```
[REPEAT COUNT]E<WIDTH>.<DECIMAL>
```

"E" INDICATES EXPONENTIAL FORMAT WITH <DECIMAL> DIGITS PRINTED TO THE RIGHT OF THE DECIMAL POINT. NOTE THAT THE EXPONENTIAL FORMAT REQUIRES THAT <WIDTH> IS >= <DECIMAL> + 7.

SPECIFIER	INTERNAL VALUE	EXTERNAL CONVERSION
E14.7	-234.5676	-0.2345676E 03

"F" FORMAT SPECIFIER

```
[REPEAT COUNT]F<WIDTH>.<DECIMAL>
```

"F" INDICATES FLOATING POINT FORMAT FORMAT WITH <DECIMAL> DIGITS PRINTED TO THE RIGHT OF THE DECIMAL POINT. NOTE THAT THE FLOATING POINT FORMAT REQUIRES THAT <WIDTH> IS >= <DECIMAL> + 3.

SPECIFIER	INTERNAL VALUE	EXTERNAL CONVERSION
F11.4	-2.34567	-2.3457

"H" FORMAT SPECIFIER

```
[REPEAT COUNT]H          (EQUIVALENT TO "F8.4")
```

THE "H" SPECIFIER INDICATES HYBRID FORMAT (EQUIVALENT TO DRM READOUT).

SPECIFIER	INTERNAL VALUE	EXTERNAL CONVERSION
2H	.2 ,1.9999	0.2000 1.9999

"I" FORMAT SPECIFIER

```
[REPEAT COUNT]I<WIDTH>
```

"I" INDICATES INTEGER FORMAT IN A <WIDTH> CHARACTER FIELD.

SPECIFIER	INTERNAL VALUE	EXTERNAL CONVERSION
I8	-234.567	-235

"N" SPECIFIER FOR "READ" STATEMENT FORMATS

```
[REPEAT COUNT]N<WIDTH>
```

THE "N" SPECIFIER INDICATES THAT THE NEXT <WIDTH> CHARACTERS WILL BE TREATED AS A NUMERIC STRING IN INTEGER, FLOATING, OR EXPONENTIAL FORMAT. THE STRING IS INTERPRETED (IGNORING BLANKS) INTIL AN ILLEGAL CHARACTER IS FOUND, OR UNTIL THE FIELD WIDTH IS EXHAUSTED.

SPECIFIER	EXTERNAL VALUE	INTERNAL CONVERSION
N8	30 45.	3045.000
N2	3045	30.00000
N7	30.5E-10	30.5E -1
2N5	1,2	1.0 , 2.0

NOTE THAT THE "N" SPECIFIER ALLOWS VERY FLEXIBLE INPUT OF DATA, AS OPPOSED TO THE FIXED FORMAT INPUT OF FORTRAN IV. <WIDTH> CAN BE SPECIFIED AS THE MAXIMUM NUMBER OF DIGITS THE USER WOULD EXPECT TO NEED. HOWEVER, IF THE ITEMS ARE SEPARATED BY ","S THE ENTIRE <WIDTH> FIELD DOES NOT NEED TO BE SPECIFIED. FOR EXAMPLE:

```
?10.10 READ(0,10.20)A,B,C
?10.20 FORMAT(N10)
?10.30 WRITE(0,10.40)A,B,C
?10.40 FORMAT(1X(1X,E11.4))
?RUN
1,2,1234567890
  0.1000E 01  0.20000E 01  0.1235E 10
IHI --WARN-- STOP AT LINE 10.40
```

THE FOLLOWING EXAMPLES ILLUSTRATE SOME TYPICAL FORMAT STATEMENTS:

```
FORMAT(1X,I10,A4,3X,I10)
FORMAT(1X(5X,E14.7))
FORMAT('THE NAME IS: '10A1)
FORMAT('0'10I8)
```

SOME ADJACENT SPECIFIERS ARE ALLOWED WITHOUT ANY DELIMITER IF THEY WOULD CAUSE NO CONFUSION IN INTERPRETATION. FOR EXAMPLE, THE FOLLOWING IS LEGAL:

```
FORMAT(' 'I10' 'I10' 4A4I10I10I10/)
```

AND THE FOLLOWING IS ILLEGAL:

```
FORMAT(1X2E10.42E10.42E10.4)
```

SINCE THE INTERPRETATION WOULD BE:

```
1X , 2E10.42 , E10.42 , E10.4  (PROBABLY NOT THE INTENDED MEANING)
```

IF YOU ARE IN DOUBT, USE A COMMA OR BLANK BETWEEN FORMAT LIST SPECIFIERS.

I/O DEVICE ASSIGNMENTS

IN IHI "READ" AND "WRITE" STATEMENTS, THE INPUT AND OUTPUT DEVICES ARE SPECIFIED BY USING UNIT NUMBERS, MUCH LIKE IN FORTRAN IV. THE FOLLOWING TABLE DEFINES THE DEFAULT ASSOCIATIONS BETWEEN IHI UNIT NUMBER AND THE SYSTEM I/O DEVICES. NOTE THAT THE SYSTEM LUNS 4-8 USED BY THE IHI PROCESSOR CORRESPOND DIRECTLY TO THE IHI UNIT NUMBERS 0-4.

IHI UNIT NUMBER	DEVICE	DEFAULT FILE SPECIFICATION
0	CONSOLE	TI0:IHI000.DAT
1	LINE PRINTER	LP0:IHI001.DAT
2	SYSTEM DISK	SY0:IHI002.DAT
3	SYSTEM DISK	SY0:IHI003.DAT
4 *	SYSTEM DISK	SY0:IHI004.DAT

* NOTE: UNIT NUMBER 4 IS USED INTERNALLY BY IHI FOR THE VIRTUAL ARRAY FILE COMMUNICATION (SEE THE SECTION ON "VIRTUAL ARRAYS" IN THIS CHAPTER). UNIT NUMBER 4 CAN NOT BE USED IN "READ" AND "WRITE" STATEMENTS AS UNIT NUMBERS 0-3 ARE USED. HOWEVER, UNIT NUMBER 4 CAN BE USED IN THE "ASSIGN" AND "CLOSE" STATEMENTS DEFINED BELOW.

THE ASSOCIATION BETWEEN THE IHI UNIT NUMBER AND DEVICE CAN BE CHANGED USING THE IHI "ASSIGN" STATEMENT WHICH IS DEFINED AS FOLLOWS:

ASSIGN STATEMENT

```
ASSIGN(<UNIT>,'<FILE SPECIFICATION>')

WHERE <UNIT> IS THE IHI UNIT NUMBER (0-4)
EXPRESSED AS A CONSTANT, VARIABLE, OR
EXPRESSION.
```

THE "ASSIGN" STATEMENT CAUSES THE NEXT "OPEN" ON IHI UNIT NUMBER <UNIT> TO OVERRIDE THE DEFAULT DEVICE ASSIGNMENT WITH THE INDICATED <FILE SPECIFICATION>. AN "OPEN" ON A PARTICULAR IHI UNIT NUMBER OCCURS THE FIRST TIME IHI READS OR WRITES TO THAT DEVICE. THE ASSOCIATION BETWEEN THE "IHI UNIT NUMBER" AND THE "DEVICE" IS MAINTAINED UNTIL A "CLOSE" STATEMENT IS EXECUTED FOR THAT UNIT NUMBER. ONCE AN "OPEN" OCCURS ON A UNIT NUMBER, ANOTHER "OPEN" WILL NOT TAKE PLACE UNTIL A "CLOSE" STATEMENT IS EXECUTED. IF AN "ASSIGN" TO AN CURRENTLY "OPEN" UNIT OCCURS PRIOR TO A "CLOSE" STATEMENT ON THAT UNIT, THEN THE RESULT OF THE "ASSIGN" WILL TAKE EFFECT AFTER THE "CLOSE" IS EXECUTED (SEE THE EXAMPLE BELOW FOLLOWING THE "CLOSE" STATEMENT).

CLOSE STATEMENT

```
CLOSE(<UNIT>)
```

WHERE <UNIT> IS AN IHI UNIT NUMBER (0-4) SPECIFIED AS A CONSTANT, INTEGER, OR EXPRESSION.

THE "CLOSE" STATEMENT CAUSES THE FILE ASSOCIATED WITH THE IHI UNIT NUMBER <UNIT> TO BE CLOSED. A "CLOSE" STATEMENT SHOULD BE EXECUTED PRIOR TO AN "ASSIGN" STATEMENT IF THE USER WISHES THE NEW ASSIGNMENT TO TAKE EFFECT ON THE NEXT REFERENCE TO THAT <UNIT> NUMBER.

THE FOLLOWING EXAMPLE ILLUSTRATES THE USE OF THE "ASSIGN" AND "CLOSE" STATEMENTS:

```
20.10 GOSUB TYPE         ;PRINT TO DEFAULT DEVICE
20.20 ASSIGN(1,'TI:')    ;ASSIGN TI: TO UNIT 1
20.30 GOSUB TYPE         ;PRINT TO DEFAULT AGAIN
20.40 CLOSE(1)           ;TERMINATE OUTPUT TO DEFAULT
20.50 GOSUB TYPE         ;NOW PRINT TO CONSOLE (TI:)
20.60 STOP

30.10 SUBR TYPE
30.20 WRITE(1,30.30)
30.30 FORMAT(' HELLO')
30.40 RETURN
?RUN
HELLO
IHI --WARN-- STOP AT LINE 20.60
```

NOTE IN THE ABOVE EXAMPLE THAT THE FIRST TWO "HELLO"'S WENT TO THE DEFAULT DEVICE ASSIOCIATED WITH IHI UNIT NUMBER 1, AND ONLY THE LAST "HELLO" WAS PRINTED ON THE CONSOLE "TI:".

READ STATEMENT ("DEFERRED" MODE ONLY)

READ(<UNIT>,<STATEMENT NUMBER>) <VARIABLE LIST>

WHERE <UNIT> IS THE IHI UNIT NUMBER (0-3) OF THE INPUT DEVICE, THE FORMAT FOR THE INPUT IS DEFINED AT <STATEMENT NUMBER>, AND <VARIABLE LIST> IS A LIST OF VARIABLE NAMES, EACH SEPARATED BY A COMMA ",".

THE "READ" STATEMENT READS IN VALUES FOR THE VARIABLES IN THE <VARIABLE LIST> FROM THE SPECIFIED <UNIT> USING THE "FORMAT" AT THE SPECIFIED <STATEMENT NUMBER>.

THE <UNIT> MUST BE A VALID IHI INPUT DEVICE (I.E. AN ATTEMPT TO READ FROM THE LINEPRINTER WILL CAUSE AN ERROR MESSAGE). SEE THE SECTION OF THIS CHAPTER ON "I/O DEVICE ASSIGNMENTS" FOR MORE DETAILS.

THE <STATEMENT NUMBER> SPECIFIED MUST BE A VALID "READ" STATEMENT FORMAT; OTHERWISE, AN ERROR MESSAGE WILL BE PRINTED. NOTE THAT IHI DIFFERS FROM FORTRAN IV IN THAT ONLY THE "A", "X", AND "N" SPECIFIERS ARE ALLOWED IN READ STATEMENT FORMATS.

BOTH THE <UNIT> AND THE <STATEMENT NUMBER> CAN BE SPECIFIED AS AS CONSTANTS, VARIABLES, OR ARITHMETIC EXPRESSIONS.

THE NUMBER OF VARIABLES IN THE <VARIABLE LIST> NEED NOT BE THE SAME AS THE NUMBER OF SPECIFIERS IN THE CORRESPONDING "FORMAT" STATEMENT. IF MORE VARIABLES ARE SPECIFIED THAN THE "FORMAT" LIST INDICATES, THEN EITHER THE ENTIRE FORMAT LIST OR ONLY THE RE-USE LIST IS RE-USED (SEE THE DESCRIPTION OF THE "FORMAT" STATEMENT FOR DETAILS).

IF AN ARRAY IS INDICATED IN THE VARIABLE LIST WITHOUT ANY SUBSCRIPTS, THEN VALUES ARE READ IN FOR THE ENTIRE ARRAY IN THE FOLLOWING ORDER (LEFT TO RIGHT, TOP TO BOTTOM):

```
ARRAY(0,0), ARRAY(1,0), ...., ARRAY(I,0),
ARRAY(0,1), ARRAY(1,1), ...., ARRAY(I,1),
    ....,        ...., ....,        ....,
ARRAY(0,J), ARRAY(1,J), ...., ARRAY(I,J)
```

THE FOLLOWING EXAMPLE READS AND WRITES A 2 BY 2 ARRAY OF VALUES:

```
?10.10  DIM A(1,1)
?10.20  READ(0,10.30)A
?10.30  FORMAT(4N8)
?10.40  WRITE(0,10.50)A
?10.50  FORMAT(1X(2I8/))
?RUN
1,2,3,4
       1       2
       3       4
IHI --WARN-- STOP AT LINE 10.50
```

WRITE STATEMENT ("DEFERRED" MODE ONLY)

WRITE(<UNIT>,<STATEMENT NUMBER>) [<LIST>]

WHERE <UNIT> IS THE UNIT NUMBER (0-3) OF THE INPUT DEVICE, THE FORMAT FOR THE OUTPUT IS DEFINED AT <STATEMENT NUMBER>, AND <LIST> IS AN OPTIONAL LIST OF VARIABLE NAMES OR ARITHMETIC EXPRESSIONS WITHIN PARENTHESES, EACH SEPARATED BY A COMMA ",".

THE "WRITE" STATEMENT WRITES OUT THE VALUES OF THE ITEMS IN THE <LIST> TO THE SPECIFIED <UNIT> IN THE FORMAT INDICATED BY THE "FORMAT" STATEMENT AT <STATEMENT NUMBER>.

THE <UNIT> MUST BE A VALID IHI OUTPUT DEVICE NUMBER AND CAN BE SPECIFIED AS A CONSTANT, VARIABLE, OR ARITHMETIC EXPRESSION. SEE THE SECTION OF THIS CHAPTER ON "I/O DEVICE ASSIGNMENTS" FOR MORE DETAILS.

THE <STATEMENT NUMBER> SPECIFIED MUST BE A VALID "WRITE" STATEMENT FORMAT. OUTPUT FORMATS REQUIRE A CARRIAGE CONTROL CHARACTER (SEE THE DESCRIPTION OF THE "FORMAT" STATEMENT FOR MORE DETAILS). IF THE <STATEMENT NUMBER> SPECIFIED IS NOT A "FORMAT" STATEMENT, OR DOES NOT EXIST, AN ERROR MESSAGE WILL BE PRINTED. THE <STATEMENT NUMBER> MAY BE SPECIFIED AS A CONSTANT, VARIABLE, OR ARITHMETIC EXPRESSION.

THE NUMBER OF ITEMS IN THE <LIST> NEED NOT AGREE WITH THE NUMBER OF CONVERSION SPECIFIERS IN THE CORRESPONDING "FORMAT" STATEMENT. IF MORE ITEMS ARE SPECIFIED THAN THE "FORMAT" LIST INDICATES, THEN EITHER THE ENTIRE LIST (OR ONLY THE RE-USE LIST) IS RE-USED. THE FOLLOWING EXAMPLE ILLUSTRATES THIS SITUATION WITH TWO DIFFERENT "FORMATS":

```
?10.10 A=1
?10.20 B=2
?10.30 C=3
?10.35 WRITE(0,10.36)
?10.36 FORMAT(' 12345678901234567890123456789012')
?10.40 WRITE(0,10.50)A,B,C,(A+B*C)
?10.50 FORMAT(1X,I10)    ;RE-USE ENTIRE FORMAT LIST
?10.60 WRITE(0,10.70)A,B,C,(A+B*C)
?10.70 FORMAT(1X(I10))  ;RE-USE "RE-USE" LIST ONLY
?RUN
12345678901234567890123456789012
         1          2          3          7
         1         2         3         7
IHI --WARN-- STOP AT LINE 10.60
```

IF AN ARRAY IS INDICATED IN THE <LIST> WITHOUT ANY SUBSCRIPTS, THEN THE ENTIRE ARRAY IS WRITTEN OUT IN THE FOLLOWING ORDER (RIGHT TO LEFT, TOP TO BOTTOM):

```
        ARRAY(0,0), ARRAY(1,0), ...., ARRAY(I,0),
        ARRAY(0,1), ARRAY(1,1), ...., ARRAY(I,1),
               ....,        ...., ....,        ....,
        ARRAY(0,J), ARRAY(1,J), ...., ARRAY(I,J)
```

THE FOLLOWING EXAMPLE PRINTS OUT A 2 BY 2 MATRIX IN THE DEFAULT CONFIGURATION (FIRST INDEX = COLUMN, SECOND INDEX = ROW):

```
?10.10 DIM A(1,1)
?10.20 A(0,0)=1
?10.30 A(1,0)=2
?10.40 A(0,1)=3
?10.50 A(1,1)=4
?10.60 WRITE(0,10.70)A
?10.70 FORMAT(1X(2I6/))
?RUN
     1     2
     3     4
IHI --WARN-- STOP AT LINE 10.70
```

FIXED FORMAT PRINT STATEMENTS

```
PRI ['<FILE SPECIFICATION>'] <EXPRESSION LIST>
PRF ['<FILE SPECIFICATION>'] <EXPRESSION LIST>
PRE ['<FILE SPECIFICATION>'] <EXPRESSION LIST>
PRH ['<FILE SPECIFICATION>'] <EXPRESSION LIST>
```

WHERE THE OUTPUT <FILE SPECIFICATION> IS OPTIONAL AND <EXPRESSION LIST> IS A LIST OF CONSTANTS, VARIABLES, OR EXPRESSIONS EACH SEPARATED BY A COMMA ",".

THE FIXED FORMAT PRINT STATEMENTS PRINT TO THE <FILE SPECIFICATION> (OR TO THE DEFAULT FILE SPECIFICATION "TI:IHISAV.IHI") THE VALUES OF A LIST OF <EXPRESSIONS>S ON SUCCESSIVE LINES IN THE FORM:

```
<SOURCE CODE OF EXPRESSION> = <VALUE>
```

WHERE <VALUE> IS PRINTED IN INTEGER, FLOATING, EXPONENTIAL, OR HYBRID FORMAT DEPENDING ON THE STATEMENT USED.

STATEMENT	OUTPUT FORMAT
PRI	I8
PRF	F11.4
PRE	E11.4
PRH	F8.4

NOTE THAT THE <FILE SPECIFICATION> CAN INDICATE A FILE ON MASS STORAGE, SUCH AS: 'DK1:PRE.IHI', HOWEVER EACH FIXED FORMAT PRINT STATEMENT WILL EITHER CREATE A NEW "VERSION" OF THE FILE SPECIFIED, OR WILL REPLACE THE SPECIFIED VERSION (IF IT EXISTS) WITH THE OUTPUT DATA FROM THE CURRENT STATEMENT. IN GENERAL, PROGRAM OUTPUT TO FILES IS BEST IMPLEMENTED USING "WRITE" STATEMENTS OR "VIRTUAL ARRAYS".

THE FOLLOWING EXAMPLE ILLUSTRATES EACH STATEMENT FORM:

```
?10.10 REAL=1.23456789
?10.11 INT=1
?10.20 PRI REAL,INT
?10.30 PRF REAL+INT,REAL-INT
?10.40 PRE 'TI:' REAL,INT
?10.50 PRH INT ,REAL   ;BLANKS CAN BE USED TO LINE ITEMS UP
?RUN
REAL =        1
INT =        1
REAL+INT =      2.2346
REAL-INT =      0.2346
REAL = 0.1235E 01
INT = 0.1000E 01
INT  =  1.2346
REAL =  1.0000
IHI --WARN-- STOP AT LINE 10.50
```

VIRTUAL ARRAYS

A FORM OF VIRTUAL ARRAY HAS BEEN IMPLEMENTED WHICH ALLOWS READ/WRITE ACCESS TO FIXED LENGTH DIRECT ACCESS FILES BY SIMPLY TYPING:

```
VIR(X) = <EXPRESSION>
```

TO WRITE TO RECORD "X" OF A VIRTUAL ARRAY FILE, OR BY USING "VIR(X)" IN AN EXPRESSION TO READ FROM A VIRTUAL ARRAY FILE:

```
PRE VIR(X)*24.3
SUM=VIR(1)+VIR(2)+VIR(3)
```

A VIRTUAL ARRAY FILE HAS A FIXED NUMBER OF TWO WORD RECORDS AND IS EQUIVALENT TO A FILE CREATED UNDER FORTRAN IV USING THE "DEFINE FILE" STATEMENT AS FOLLOWS:

```
DEFINE FILE 1(NREC,2,U,I)
```

WHERE "NREC" IS THE NUMBER OF RECORDS IN THE FILE

VIRTUAL ARRAY FILES CAN BE VISUALIZED AS SINGLE DIMENSION REAL ARRAYS AND ARE PHYSICALLY STRUCTURED AS FOLLOWS:

```
                     !----------!----------!
   RECORD 1          ! WORD 1   ! WORD 2   !
                     !----------!----------!
   RECORD 2          ! WORD 1   ! WORD 2   !
                     !----------!----------!
                        ....       ....
                        ....       ....
                     !----------!----------!
   RECORD "NREC"     ! WORD 1   ! WORD 2   !
                     !----------!----------!
```

A VIRTUAL ARRAY FILE CAN BE CREATED FROM IHI BY SIMPLY TYPING:

```
VIR(0) = NREC
```

WHERE "NREC" IS THE NUMBER OF RECORDS IN THE FILE

THE FILE NAME IS THE SPECIFICATION ASSOCIATED WITH IHI UNIT NUMBER 4 (SEE THE SECTION OF THIS CHAPTER ON "I/O DEVICE ASSIGNMENTS") AND CAN BE CHANGED USING THE "ASSIGN" STATEMENT. OUTPUT TO A PARTICULAR FILE CAN BE TERMINATED USING THE "CLOSE(4)" STATEMENT.

IF THE USER ATTEMPTS TO CREATE ANOTHER VIRTUAL ARRAY FILE PRIOR TO CLOSING AN "OPEN" FILE ON DEVICE 4, THEN IHI WILL AUTOMATICALLY "CLOSE" THE FILE AND OPEN A NEW VIRTUAL ARRAY FILE. IF THE FILE NAME HAS NOT BEEN CHANGED VIA AN "ASSIGN" STATEMENT, THEN A NEW VERSION OF THE FILE CURRENTLY ASSOCIATED WITH IHI UNIT NUMBER 4 IS CREATED.

ONCE A FILE HAS BEEN CREATED, ITS LENGTH (NUMBER OF RECORDS) CAN NOT BE MODIFIED. THE USER CAN DETERMINE THE LENGTH OF THE CURRENT OPEN VIRTUAL ARRAY FILE BY SIMPLY READING RECORD 0. FOR EXAMPLE:

```
                NREC=VIR(0)
        OR      PRI VIR(0)
```

NOTE: RECORD 0 DOES NOT PHYSICALLY EXIST IN THE VIRTUAL ARRAY FILE.

THE FOLLOWING IS AN EXAMPLE FORTRAN IV PROGRAM TO CREATE A 100 RECORD VIRTUAL ARRAY FILE ("SY0:FOR001.DAT" BY DEFAULT) AND FILL IT WITH VALUES OF THE SINE FUNCTION:

```
                EXAMPLE FORTRAN IV PROGRAM
                --------------------------

                NREC=100
                DEFINE FILE 1(NREC,2,U,INDEX)
                DO 10 I=1,NREC
                R=I
                R=SIN(R)
        10      WRITE(1'I)R
                END
```

AND THE FOLLOWING IS AN EXAMPLE IHI PROGRAM WHICH READS FROM THE FILE "SY0:FOR001.DAT" AND COMPARES THE IHI "SIN" FUNCTION WITH THE FORTRAN IV "SIN" FUNCTION AND PRINTS ANY DIFFERENCES (NONE FOUND):

```
                EXAMPLE IHI PROGRAM
                -------------------

                ?10.10 ASSIGN(4,'FOR001')
                ?10.30 FOR I=1,VIR(0)
                ?10.40 WHEN ( SIN(I) 'NE' VIR(I) ) PRE I,SIN(I),VIR(I)
                ?10.50 NEXT I
                ?10.60 CLOSE(4)
                ?RUN
                IHI --WARN-- STOP AT LINE 10.60

                ?
```

CHAPTER 8
ERROR MESSAGES

IHI ERROR MESSAGES ARE STORED IN TWO FILES ON THE SYSTEM DISK. THESE FILES CHANGE FROM TIME TO TIME AS IHI IS UPGRADED AND DEPENDING ON THE SYSTEM. THUS THE USER SHOULD REFER TO THESE FILES FOR A DETAILED LIST OF ERROR MESSAGES. THE ERROR MESSAGES ARE SELF EXPLANATORY AND WHEN THEY ARE PRINTED MOST ERROR MESSAGES ALSO INCLUDE A STRING OF ASCII CODE WHICH INDICATES WHAT CAUSED THE ERROR TO OCCUR. IN THE MESSAGE FILES:

%VA INDICATES THAT A VARIABLE LENGTH STRING OF SOURCE CODE WILL FOLLOW ON THE SAME LINE AS THE ERROR MESSAGE.

%L%VA INDICATES THAT A VARIABLE LENGTH STRING OF SOURCE CODE WILL FOLLOW ON THE NEXT LINE AFTER THE ERROR MESSAGE.

THE ERROR MESSAGES CONTAINED IN THE SYSTEM DISK FILE: "SY:IHIINS.MSG" EITHER INDICATE ERRORS IN THE INPUT <COMMAND STRING> TO IHI OR ERRORS WHICH ARE DISCOVERED IN THE IHI "IMMEDIATE" MODE WHEN A STATEMENT IS FIRST INSERTED BY THE USER, AND IHI IS PERFORMING THE PASS 1 TRANSLATION TO THE PACKED BINARY INTERNAL FORM.

THE ERROR MESSAGES CONTAINED IN THE SYSTEM DISK FILE: "SY:IHIRUN.MSG" INDICATE ERRORS WHICH OCCUR AT RUN TIME DURING THE EXECUTION OF EITHER "IMMEDIATE" OR "DEFERRED" MODE IHI STATEMENTS.

APPENDIX A
SUMMARY OF IHI COMMANDS AND STATEMENTS

* INDICATES STATEMENTS WHICH ARE EXECUTABLE IN THE "DEFERRED" MODE ONLY.
** INDICATES LEGAL PROMPT MODE <VERB>S (SEE "PRMT" COMMAND).

```
      COMMAND            MEANING AND SYNTAX
      -------            ------------------

      @<FILE SPEC>       INDIRECT COMMAND FILE SPECIFICATION

      RUN                RUN THE CURRENT STORED IHI PROGRAM

   ** LIST               LIST CURRENT STORED IHI PROGRAM:
                           LIST ['<FILE SPEC>'] [ GG[.LL] [,GG[.LL]] ]

   ** DELETE             DELETE ONE OR MORE STORED IHI STATEMENTS:
                           DELETE  GG[.LL] [,GG[.LL]]

      ZAP                DELETE ALL CORE BLOCKS ALLOCATED BY THE
                         EXECUTION OF IHI STATEMENTS

      RENEW              DELETE ALL STORED STATEMENTS AND INITIALIZE
                         ALL ALLOCATED CORE BLOCKS. EQUIVALENT TO:
                         "DELETE 1,99" FOLLOWED BY "ZAP".

   ** SAVE               SAVE BINARY CORE IMAGE OF STORED STATEMENTS:
                           SAVE ['<FILE SPEC>'] [ GG[.LL] [,GG[.LL]] ]
                           (DEFAULT FILE IS "SY:IHISAV.IHI")

      LOAD               LOAD BINARY CORE IMAGE OF IHI STATEMENTS THAT
                         WERE SAVED BY "SAVE":
                           LOAD ['<FILE SPEC>']
                           (DEFAULT FILE IS "SY:IHISAV.IHI")

   ** OVRLAY             OVRLAY WORKS LIKE "LOAD" BUT DOES NOT DELETE
                         STORED STATEMENTS PRIOR TO LOADING; NEW
                         STATEMENTS SUPERSEDE OLD STATEMENTS.

      CON                CONTINUE AT STATEMENT FOLLOWING "PAUSE"
                         (VALID ONLY AFTER A "PAUSE")

      PRMT <VERB>        ENTER PROMPT MODE. IHI PROMPTS WITH "?VERB "
                         (LEGAL PRMT <VERB>S ARE INDICATED BY **).
   STATEMENT        MEANING AND SYNTAX
   ---------        ------------------

   NOP              CAUSES NO OPERATION

   DIM              DIMENSION A LIST OF ARRAYS:
                      DIM VAR1(I1[,J1]), VAR2(I2[,J2]), ...

** LET              ASSIGNMENT STATEMENT ("LET" IS OPTIONAL):

   JUMP             JUMP TO STATEMENT OR GROUP:
                      "JUMP GG.LL" OR "JUMP GG"

 * WHEN             CONDITIONAL STATEMENT:
                      WHEN (<EXP><RELATION><EXP>) <STATEMENT>

 * FOR-NEXT         LOOP CONTROL STATEMENTS:
                      FOR <VAR>=<EXP1>,<EXP2>[,<EXP3>]
                      NEXT <VAR>

 * PAUSE            PAUSE PROGRAM AND RETURN TO "IMMEDIATE" MODE
                    (TO CONTINUE TYPE "CON")

 * STOP             STOP PROGRAM AND RETURN TO "IMMEDIATE" MODE

** CALL             CALL A PRE-DEFINED IHI SYSTEM SUBROUTINE (HCR):
                      CALL <SUB NAME>(<ARG LIST>)

   GOSUB <NAME>     TRANSFERS CONTROL TO THE DEFERRED MODE IHI
                    SUBROUTINE <NAME>.

 * SUBR <NAME>      INDICATES THE START OF AN IHI SUBROUTINE

 * RETURN           RETURNS CONTROL TO THE STATEMENT FOLLOWING
                    THE LAST "GOSUB" STATEMENT.

 * FORMAT           DEFINES HOW THE FORMATTED ASCII RECORDS ARE TO
                    BE USED WITH THE CORRESPONDING "READ" OR "WRITE"
                    STATEMENTS.

** ASSIGN           ASSIGNS AN IHI UNIT NUMBER TO A CORRESPONDING
                    FILE SPECIFICATION:
                      ASSIGN(<UNIT>,'<FILE SPECIFICATION>')

** CLOSE            CLOSES THE FILE ASSOCIATED WITH THE SPECIFIED
                    IHI UNIT NUMBER:
                      CLOSE(<UNIT>)
```

	STATEMENT	MEANING AND SYNTAX
*	READ	READS IN THE VALUES FOR A LIST OF VARIABLES FROM THE SPECIFIED <UNIT> ACCORDING TO THE FORMAT AT THE INDICATED <STATEMENT NUMBER>: READ(<UNIT>,<STATEMENT NUMBER>)<VARIABLE LIST>
*	WRITE	WRITES OUT THE VALUES OF THE ITEMS IN THE LIST TO THE SPECIFIED <UNIT> USING THE FORMAT AT THE INDICATED <STATEMENT NUMBER>: WRITE(<UNIT>,<STATEMENT NUMBER>)<LIST>
**	PRI	PRINT <EXPRESSION LIST> USING "I8" FORMAT PRI ['<FILE SPEC>'] <EXPRESSION LIST>
**	PRF	PRINT <EXPRESSION LIST> USING "F11.4" FORMAT PRF ['<FILE SPEC>'] <EXPRESSION LIST>
**	PRE	PRINT <EXPRESSION LIST> USING "E11.4" FORMAT PRE ['<FILE SPEC>'] <EXPRESSION LIST>
**	PRH	PRINT <EXPRESSION LIST> USING "F8.4" FORMAT PRH ['<FILE SPEC>'] <EXPRESSION LIST>
*,	PRO	

APPENDIX B
AD/5 ANALOG DEVICE MNEMONICS

READABLE ANALOG DEVICE MNEMONICS	MEANING
ADC	ADC-MUX CHANNEL READOUT (BY PATCHBOARD ADDRESS)
AMP	ANALOG AMPLIFIER OUTPUT
DER	AMPLIFIER OUTPUT IN "TEST" MODE SCALED TO EQUAL THE DERIVATIVE INPUT IF THE AMPLIFIER IS AN INTEGRATOR.
POT	OUTPUT OF A POTENTIOMETER IN THE "COEF*INPUT" MODE
DCU	OUTPUT OF A DCU IN THE "COEF*INPUT" MODE (DCU'S ARE INVERTING COMPONENTS)
COF	COEFFICIENT SETTING OF A POT OR DCU (READBACK OF DCU SETTING IS INVERTED)
TRK	ADDRESSABLE ANALOG TRUNK
DAC	OUTPUT OF A MULTIPLYING DAC (SETTING TIMES MINUS THE DAC INPUT)
NLN	NON-LINEAR
MSC	MISCELLANEOUS

SETTABLE ANALOG DEVICE MNEMONICS	MEANING (LEFT OF "=")
COF	SET A POTENTIOMETER OR DCU
DAC	SET A DAC

APPENDIX C
AD/5 IHI CALLABLE SUBROUTINES

THIS APPENDIX DEFINES THE HCR'S AND SUBROUTINES WHICH HAVE BEEN IMPLEMENTED IN IHI. THEY ARE THE SAME HCR'S WHICH ARE CALLABLE FROM FORTRAN IV AND MACRO AS DISCUSSED IN CHAPTER 4 OF THE "SYSTEM 511 HYBRID USER'S MANUAL". THE ROUTINES CAN BE CALLED USING THE IHI "CALL" STATEMENT AS DEFINED IN CHAPTER 7 OF THIS MANUAL.

TO USE ANY HYBRID COMMUNICATION ROUTINES, THE USER MUST BE "ATTACHED" TO THE ANALOG CONSOLE(S) WITH WHICH HE PLANS TO COMMUNICATE. REFER TO THE DEFINITION OF THE IHI <COMMAND STRING> SPECIFICATION IN CHAPTER 2.

IN IHI THE HCR'S ARE CALLED WITH HCR ERROR TESTING ENABLED (REFER TO THE DEFINITION OF "HYTST" IN CHAPTER 4 OF THE "SYSTEM 511 HYBRID USERS MANUAL").

THE FOLLOWING IHI ERROR MESSAGES CAN OCCUR IF AN ERROR CONDITION IS DISCOVERED BY THE HCR:

```
IHI **FATL** HCR TYPE 0 (ADDR) ERROR
IHI **FATL** HCR TYPE 1 (DATA) ERROR
IHI **FATL** HCR TYPE 2 (BUSY) ERROR
IHI --WARN-- HCR TYPE 3 (POT SET) ERROR
```

IN IHI ALL VARIABLES ARE REAL VARIABLES. THEREFORE, IHI MUST CONVERT THE IHI ARGUMENTS TO THE INTEGER FORMAT REQUIRED BY THE HCR'S. THE HCR'S WHICH READ ANALOG ADDRESSES AND SET ANALOG COEFFICIENT DEVICES INHERENTLY EXPECT INTEGER ARGUMENTS SCALED SUCH THAT 10000 IS EQUIVALENT TO ANALOG REFERENCE (1.0 ON THE DRM READOUT). IN IHI, SCALED ARGUMENTS HAVE BEEN IMPLEMENTED IN SOME HCR'S SO THAT ANALOG READOUTS AND COEFFICIENT SETTINGS ARE SCALED SUCH THAT 1.0 IN IHI IS EQUIVALENT TO ANALOG REFERENCE. THIS CORRESPONDS TO THE SCALING USED FOR THE IHI ANALOG DEVICE MNEMONICS.

THE FOLLOWING CONVENTIONS ARE USED IN SPECIFYING THE ARGUMENTS OF THESE SUBROUTINES:

IV = <VARIABLE> WHICH WILL CONTAIN THE REAL EQUIVALENT OF THE INTEGER VALUE RETURNED BY THE HCR.

SV = <VARIABLE> WHICH WILL CONTAIN THE SCALED EQUIVALENT OF THE INTEGER VALUE RETURNED BY THE HCR (I.E. THE SCALED EQUIVALENT OF 10000 IS 1.0).

IE = <EXPRESSION> WHICH WILL BE CONVERTED BY IHI TO THE NEAREST INTEGER VALUE, AS REQUIRED BY THE HCR.

SE = <EXPRESSION> SCALED SUCH THAT 1.0 IS EQUIVALENT TO THE INTEGER VALUE 10000 (WHICH THE HCR NORMALLY EXPECTS).

TWO GENERAL SUBROUTINES, "AD5RD" AND "AD5WT", HAVE BEEN IMPLEMENTED IN IHI TO ALLOW BASIC HARDWARE LEVEL COMMUNICATION WITH THE SYSTEM 511 HYBRID INTERFACE. A CONTROL OPERATION CONSISTS OF COMMUNICATING AN ADDRESS AND A DATA WORD. THE INFORMATION NECESSARY TO USE THESE ROUTINES IS FOUND IN CHAPTER 6 OF THE "SYSTEM 511 HYBRID USERS MANUAL"; THE USER SHOULD BE THOROUGHLY FAMILIAR WITH THIS MATERIAL PRIOR TO USING THESE SUBROUTINES. HYBRID COMMUNICATIONS NOT COVERED BY THE OTHER HCR'S CAN BE PERFORMED USING THESE ROUTINES; THE ARGUMENTS OF THESE ROUTINES INDICATE THE CONTROL ADDRESS AND DATA WORD.

SUBROUTINE — FUNCTION PERFORMED

AD5RD(IE,IV)

READ OPERATION FROM THE AD-5 HYBRID INTERFACE
IE = DECIMAL EQUIVALENT OF THE HIF/DAP OCTAL ADDRESS.
IV = VALUE RETURNED BY THE READ OPERATION (THE DECIMAL EQUIVALENT OF THE 16 BIT DATA WORD RETURNED BY THE INTERFACE).

FOR EXAMPLE:

```
CALL AD5RD(3,DRM)
```

RETURNS CONSOLE 0 DRM OUTPUT IN "DRM"

IF THE ANALOG CONSOLE IS NOT "ATTACHED", A FATAL ERROR MESSAGE WILL OCCUR AND IHI WILL RETURN TO THE "IMMEDIATE" MODE.

AD5WT(IE1,IE2)

WRITE OPERATION TO THE AD-5 HYBRID INTERFACE
IE1 = DECIMAL EQUIVALENT OF THE HIF/DAP OCTAL ADDRESS.
IE2 = DATA WORD TO BE WRITTEN TO THE AD-5 (DECIMAL EQUIVALENT OF THE VALUE OF THE 16 BIT WORD).

FOR EXAMPLE:

```
CALL AD5WT(0,68)
```

IS THE CONSOLE CONTROL COMMAND WRITE (OCTAL DATA WORD: 000104) WHICH PLACES THE ANALOG CONSOLE IN THE "RUN" MODE.

THE ANALOG CONSOLE MUST BE "ATTACHED", AS DISCUSSED IN THE "AD5RD" SUBROUTINE.

```
SUBROUTINE             FUNCTION PERFORMED
----------             ------------------

AUTHD(IE)              IE = 0 TURNS AUTO-HOLD "OFF"
                          = 1 TURNS AUTO-HOLD "ON"

CONSO(IE)              SELECT ANALOG CONSOLE NUMBER "IE" TO
                       TO ALLOW HCRS AND ANALOG DEVICE MNEMONICS
                       TO COMMUNICATE WITH CONSOLE NUMBER "IE".
                       (NOTE: CONSOLE "IE" MUST BE "ATTACHED")

HOFF(IE)               DISABLE HYBRID ACCESS TO ANALOG CONSOLE "IE"
                       (TURN "HYB ON" SWITCH OFF)

HOLD                   PUT ANALOG CONSOLE IN THE "HOLD" MODE

HON(IE)                ENABLE HYBRID ACCESS TO ANALOG CONSOLE "IE"
                       (TURN "HYB ON" SWITCH ON)

HYTOL(SE)              SET TOLERANCE FOR "HCR TYPE 3" ERRORS TO "SE"

IC                     PUT ANALOG CONSOLE IN THE "IC" MODE

INITA(IE)              INITIALIZE ANALOG CONSOLE "IE" AS FOLLOWS:
                       (NOTE: CONSOLE "IE" MUST BE "ATTACHED")
                                 TIME SCALE - X1
                                 ANALOG MODE - IC
                                 LOGIC MODE - LOAD
                                 INTERVAL TIMER USES
                                    THUMB WHEELS
                                 1 SEC. V SIGNAL
                                 ALL OTHER PUSHBUTTONS "OFF"
                                 MDACS UPDATE IMMEDIATE
                                 ADC CHANNELS TO SAMPLE
                                 CONTROL LINES TO 0
                                 SENSE LINES TO 0

LEX(IE)                IE = 0 TURNS "LOGIC EXECUTE" SWITCH "OFF"
                       IE = 1 TURNS "LOGIC EXECUTE" SWITCH "ON"

LODE                   PUT ANALOG CONSOLE IN "LOAD" MODE

LRUN                   PUT ANALOG CONSOLE IN THE LOGIC "RUN" MODE

LSTOP                  PUT ANALOG CONSOLE IN THE LOGIC "STOP" MODE

OP                     PUT ANALOG CONSOLE IN THE "OPERATE" MODE

PB(IE)                 IE = 0 TURNS "PB" SWITCH "OFF"
                          = 1 TURNS "PB" SWITCH "ON"
```

SUBROUTINE	FUNCTION PERFORMED
READ(IE.SV)	READ THE ANALOG ADDRESS "IE" AND RETURN THE SCALED VALUE IN "SV" (1.0 IS EQUIVALENT TO ANALOG REFERENCE)
READH(IE.SV)	READ ADC-MUX CHANNEL NUMBER "IE" AND RETURN THE SCALED VALUE IN "SV"

SELIT(IE)	IE = 0 SPECIFIES THAT THE INTERVAL TIMER PERIODS WILL BE CONTROLED BY THE THUMB WHEEL SWITCHES
	IE = 1 SPECIFIES THAT THE INTERVAL TIMER PERIODS WILL BE CONTROLED BY THE INTERVAL TIMER REGISTER

SELVS(IE)	IE = 0 SELECTS 1 SEC. V SIGNAL = 1 SELECTS 100 MS V SIGNAL = 2 SELECTS 10 MS V SIGNAL
SENSR(IV)	RETURN THE VALUE OF THE SENSE LINE REGISTER IN "IV"
SETAR(IE)	SET THE ANALOG ADDRESS "IE" INTO THE ANALOG ADDRESS REGISTER
SETC(IE)	SET CONTROL CONDITION DETERMINED BY "IE" INTO CURRENT ANALOG CONSOLE (REFER TO THE TABLE IN THE HCR MANUAL DEFINING THE OPERATION OF "SETC" AS A FUNCTION OF "IE").
SETCR(IE)	SET CONTROL LINE REGISTER TO THE INTEGER VALUE OF THE EXPRESSION "IE"
STEP	IF THE LOGIC MODE ON THE CURRENT CONSOLE IS "STOP", THE LOGIC IS ADVANCED THROUGH ONE CLOCK PERIOD
STIND(IE.SE)	SET THE COEFFICIENT DEVICE WITH THE ADDRESS "IE" TO THE SCALED VALUE "SE" (SE=1.0 TO SET A COEFFICIENT TO 1.0)
STINH(IE1.SE.IE2)	SET DAC/DCU NUMBER "IE1" TO THE VALUE OF THE SCALED EXPRESSION "SE" USING UPDATE CODE "IE2"

```
SUBROUTINE           FUNCTION PERFORMED
----------           ------------------

STITR(IE1,IE2,IE3)   SET THE PERIODS OF THE INTERVAL TIMER
                     REGISTER AS FOLLOWS:
                        A PERIOD = "IE1" COUNTS
                        B PERIOD = "IE2" COUNTS
                        C PERIOD = "IE3" COUNTS

STREF(IE)            IE = 0 TURNS "COEF X IN" SWITCH "OFF"
                            (COEFFICIENT SETTINGS READ)
                     IE = 1 TURNS "COEF X IN" SWITCH "ON"
                            (COEFFICIENT DEVICES READ THEIR
                              SETTING TIMES THEIR PATCHED INPUT)

TEST(IE)             IE = 0 TURNS "TEST" SWITCH "OFF"
                        = 1 TURNS "TEST" SWITCH "ON"

TSCAL(IE)            SELECT ANALOG TIME SCALE AS FOLLOWS:

                        IE      TIME SCALE
                        --      ----------
                         0          X1
                         1          X10
                         2          X100
                         3          X1000

VER(IE)              IE = 0 TURNS "PROB VER" SWITCH "OFF"
                        = 1 TURNS "PROB VER" SWITCH "ON"
```