# SECTION 8
# CCIL LANGUAGE MANUAL

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

# SECTION 1. INTRODUCTION

# SECTION 1. INTRODUCTION

## CONCEPTS

### Purpose

This manual defines and illustrates the computer operations associated with the Control Center Interactive Language (CCIL), a subset of the IUE Control Center operational software system. This manual assumes that the reader has no previous computer programming experience, although from time to time reference will be made to computer terms for the benefit of those who do have some programming background.

CCIL is a high-level computer language, that is, more like the FORTRAN language than an assembly language. This means that instructions to the computer are input in a form that approximates an English language sentence rather than a pure string of numbers. Incorporated in CCIL are instructions for monitoring telemetry, for requesting commands to be sent, for specifying the display or print of information for documentation purposes, and for control of sets of instructions called procedures.

Confusion is sometimes caused because this manual is designed to be used primarily by procedure writers and reviewers, and because the predecessor of CCIL was called the Procedure Control Language (PCL). However, almost all of the instruction set of this language may be used as single computer requests issued from the keyboard, in addition to being used as a pre-defined combination of requests that are stored in the computer, called a procedure. Furthermore, CCIL has a

much larger instruction set than PCL and, therefore, is capable of performing many more tasks than will be discussed in this manual. For a full discussion of all CCIL capabilities, such as maneuvering, data base maintenance, and experiment display, refer to the latest Computer Science Corporation "Control Center Software System Operations Manual" (CSC/SD-76/6055, IM I-76-109).

## Organization

This manual is informally divided into two parts. Sections 1 and 2 discuss concepts and background information necessary for proper utilization of the CCIL. The remainder of the manual discusses the instructions (directives) and gives examples of their use. Appendixes and an index are also included. The index will provide multiple references to information not thoroughly covered as a single topic in the text. Therefore, to thoroughly understand a subject (subroutines for instance), look in the index for the entry to find other references to the subject. Of course, the index shows all CCIL directives in alphabetical order so there is no need to search the Table of Contents.

## IUE System

Figure 1-1 shows the basic computer system used for IUE control. Directives are available for transferring information to and from the various devices shown in the figure. Analyst interaction is primarily with the CRT keyboard and display and with the stripchart recorders. Any requested printer or magnetic tape output is usually brought in by computer operations personnel.

Commands and individual (non-procedure) requests are made from the CRT keyboard. There are also written procedures which are loaded into the system from magnetic tape. These procedures are executed by typing:

EXEC (procedure name), (arguments).

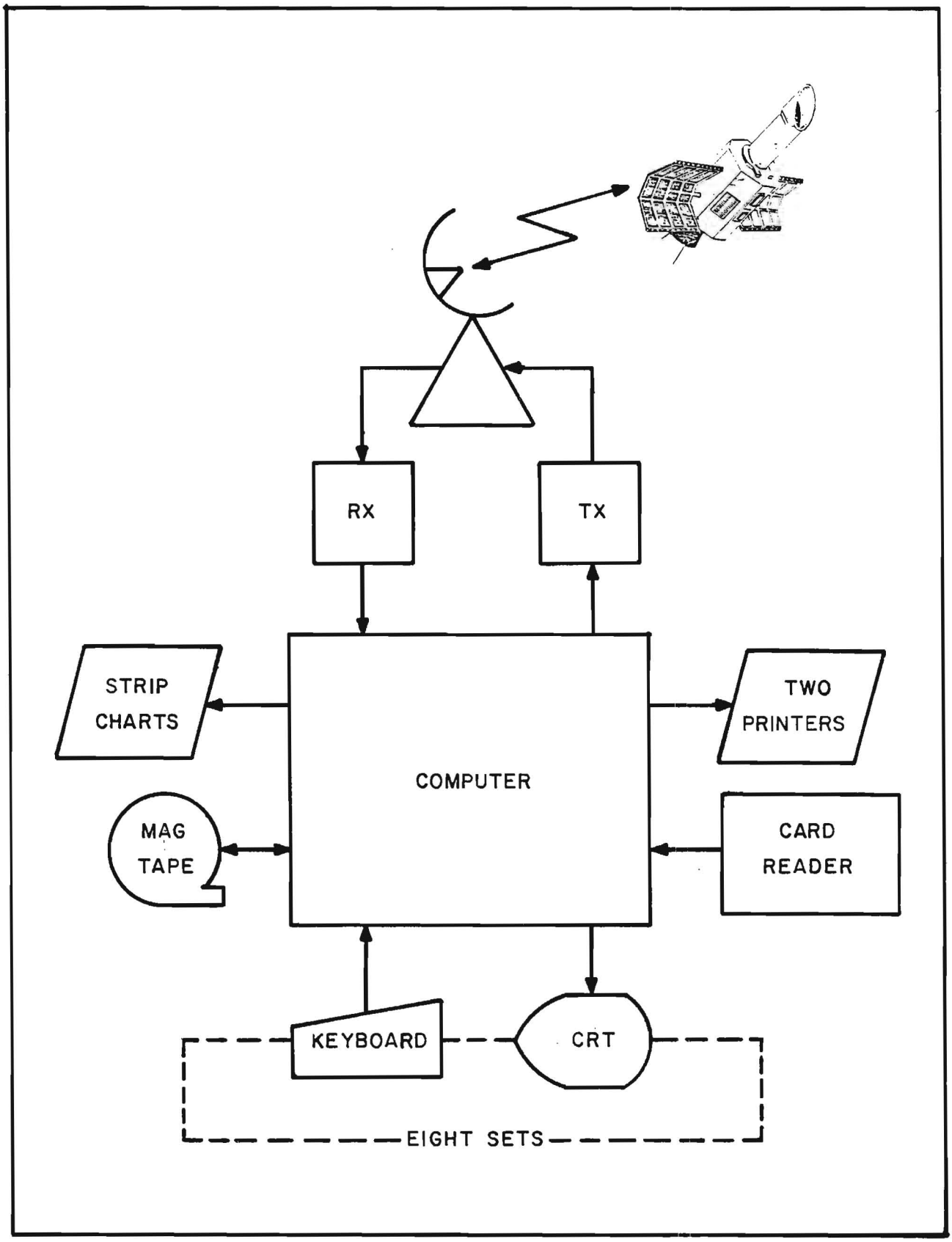



Figure 1-1. IUE System

## Operational Modes

CCIL has been divided into units, called operational modes, to distinquish the separate uses of the instruction set. These modes are:

a. Procedure
b. Single-statement
c. Suspended-procedure

The Procedure mode generally uses the complete instruction set, while the other two modes use approximately the same subset of the total instruction capabilities. Each mode always has a defined set of legal directives. The Procedure mode uses all directives except the directive in Section 4 under the heading SUSPENDED PROCEDURE ONLY, the GO directive. The single-statement mode uses all directives except those in Section 4 under the headings PROCEDURE ONLY, PROCEDURE AND SUSPENDED PROCEDURE, and SUSPENDED PROCEDURE ONLY. Finally, the Suspended procedure mode uses all directives except those in Section 4 under the heading PROCEDURE ONLY.

While this is a reasonable basis for dividing CCIL into smaller units, too much emphasis should not be placed on this concept. There are other operating characteristics which are just as, if not more, important as available directives. One might just as well say that there are only two modes, Single-statement and Procedure. The Suspended procedure mode is operationally a form, or subset, of the Procedure mode.

One can operationally go back and forth between the Single-statement and Procedure modes and go back and forth between the Procedure and Suspended procedure modes, but one cannot go back and forth between Single-statement and the Suspended procedure modes. Additionally, a family of variables (refer to Section 2) is passed between the Procedure and the Suspended procedure modes, but the Single-statement mode has a unique set of variables.

## Procedures

Another confusing term in the CCIL vocabulary is "procedure." Often procedure is used to mean the conglomeration of whatever is necessary to run some operation on the computer; e.g., Camera Test Procedure. This meaning is synonymous with the words "task" or "operation." For CCIL purposes, this is not a precise enough definition because it can include more than one actual CCIL procedure and it can include one or more CCIL subroutines.

Specifically, a CCIL procedure is a set of statements that begins with a PROC statement (an acronym for PROCedure) and ends with a PEND statement (an acronym for Procedure END). The procedure's set of statements cannot include any other PROC or PEND, which means a procedure cannot contain another procedure or a subroutine. Procedures may contain references to other procedures and subroutines by containing specific directives which cause them to be executed or run.

Procedures must have a unique label on the PROC statement that specifies the name of the procedure. Labels, other than the PROC label, can be coded into a procedure as a prefix on any directive. These labels need to be unique within a procedure, but need not be unique in the total system. That is, they may be duplicated in other procedures and subroutines without causing any problems. Labels are used in conjunction with the GOTO, IF, DO, HOLD, and GO directives. They may also be used without any functional purpose, as headings for place marking or for ease of reading.

Entry into a procedure, through the issuing of an EXEC directive, can be accompanied by the passing of up to eight arguments*. The value of these arguments are stored in an AF array, and are accessed by the procedure by reference to AF(i), "i" being a value one to eight. A procedure may be entered from either the keyboard (Single-statement mode), from another procedure, or from a subroutine.

* See Index for reference to other discussion of this topic.

Local variables* may be defined by a procedure. The names of these variables can duplicate local variable names in other procedures. They should not duplicate names used for any global variables or names used for any local variables in subroutines which a procedure references. Further explanation of this concept is given under the heading "Environment."

## Subroutines

A CCIL subroutine is a set of statements that begins with a SUBR statement and ends with a PEND statement. The set cannot contain any other SUBR, PROC, or PEND statements; that is, it cannot physically contain any other subroutine or procedure. However, subroutines may contain directives which reference other procedures and subroutines.

A unique label is also prefixed to the SUBR directive to generate the subroutine name. This means the name cannot be the same as a name used on a SUBR directive for any other subroutine. It can, however, be the same as a name of a procedure, since the PROC and SUBR directives are handled differently.

Labels, other than a SUBR label, are handled in the same manner as discussed in the paragraph under the Procedures heading. They also must be unique within the subroutine.

Entry into a subroutine is made by issuing a CALL directive. Up to eight arguments can be passed; the value of these arguments are stored in an array named ARG. They are accessed by the subroutine by reference to ARG(i), "i" being a value one to eight. Subroutines can be called from a procedure or from another subroutine but not from Single-statement mode.

Subroutines may also reference the AF argument array of the procedure that was being executed before the subroutine was called.

* See Index for reference to other discussion of this topic.

Local variables may also be defined by a subroutine. The name of the variable can duplicate a local variable name in other subroutines. It should not duplicate names used for global variables or names used for local variables in the procedure which called the subroutine. Refer to the Environment heading for further discussion.

## Modules

The previous headings have shown some of the differences between procedures and subroutines. In some aspects, they have identical characteristics. Rather than repeating the words "procedures and subroutines" when referring to the similarly handled concepts, the terms "module" or "routine" will be used to mean either a subroutine or a procedure.

All modules have single entry points, the label, which defines the module. Once entered, the module does not have to execute in sequential order from top to bottom, it may execute in any path.

Multiple RETURN directives can be coded into routines to provide alternate exit paths. If no RETURN is used, the routine may still exit through the PEND statement.

When writing control center operational routines, a series of statements can take the form of either a single procedure or can be a procedure connected with variable numbers of other procedures and subroutines. Initial coding should strive for straightforward simplicity, not for complicated sophistication. Unless multiple modules are needed, stick to a single-procedure operation to avoid the myriad of problems associated with multiple modules.

The obvious question arises: when are multiple modules needed? A few examples will illustrate the concept. Let's suppose a routine is being written for a device and this routine requires a particular configuration of the device before the routine can execute. One

option is to code the configuration statements on the front of the procedure, or, if available, to execute an existing procedure or subroutine that currently performs the configuration task. The second alternative implies that the contents of the current computer-resident group of routines is known. If the contents are known and if the variable names used will cause no problem, then use the existing procedure to do some of your work.

Calling subroutines may have the effect of reducing the total physical size of the routine. If a set of statements are duplicated several times throughout a procedure, they might be removed and coded once as a subroutine. Then a single CALL statement in the procedure may be used rather than the set of statements. It is also possible to write a more structured procedure by placing sets of detailed code in subroutines and then having the procedure call these structures.

## FORMATS

### Character Set

All characters are in Ø29 card punch format. The following characters may be used in creating label or variable names:

| | |
|---|---|
| A-Z | capital letters |
| Ø-9 | decimal digits |

The remainder of the character set are characters that have special meaning to the operational system. They cannot be used in labels or variable names. They may, with the exception of the single quote and the semicolon, be used in text output, that is, character strings between single quote marks.

| Character | Description | CCIL Function |
|---|---|---|
| * | asterisk | In card column 1, start of comment field; otherwise multiplication |

| Character | Description | CCIL Function |
|---|---|---|
| | blank | Delimeter, or separation between operational fields |
| , | comma | Delimeter between argument fields |
| = | equal | Equality |
| . | decimal point | Delimeter for logical operations or as numerical decimal placement |
| ( | left parenthesis | Start of index field for array or altered precedence of operation |
| ) | right parenthesis | End of index field for array or an altered precedence of operation |
| ' | single quote | Start and end of text definition (character string for display or print); or start and end of special input parameters |
| + | plus | Addition |
| - | minus | Subtraction or 2's complement |
| / | slash | Start of comment or division |
| ; | semicolon | End of record (CCIL); end of executable function (PCL) |
| : | colon | Start of command field |
| & | and | Logical and |
| \| | vertical bar | Logical inclusive or |
| ¬ | not | Unary not or 1's complement |
| > | right bracket | Relational greater than |
| < | left bracket | Relational less than |

Any other characters found on a Ø29 card punch may be used in text string only.

## CCIL Statement Format

A CCIL statement is a string of characters consisting of four fields. If a semicolon is encountered, the statement is terminated. Its fields, from left to right, are:

### Label Field (optional in procedure mode, illegal in others)

a. Starts in column 1 with an alphabetic character.
b. One to eight characters in length.
c. Contains only alphabetic characters or decimal digits.
d. Terminates with a blank or a semicolon.

### Directive Field (required)

a. Starts in column 2 or beyond.
b. Is either a:
   (1) Directive defined in this manual, or
   (2) A defined variable.
c. Terminates with a blank, an equal sign, or a semicolon.

### Argument Field (dependent on directive)

a. May contain multiple arguments, separated by commas.
b. Construction of arguments is defined by the particular operation.
c. Terminates with a blank, column 72, or a semicolon.

### Comments Field (optional)

a. Starts with a slash (/) character.
b. Terminates with column 72 or semicolon.

The convention of this manual will be to limit label length to six characters and to start the directive field in column 8. This is done only for cosmetic purposes. The printout, or listing, of the constructed procedure is easier to read; however, any spacing format may be used as long as the above rules are followed. Cards must be punched on an Ø29 machine.

For example:

```
1       8 (columns)
TESTIT  EXEC  TURNON,5,SYSTEM,2.7  /SYSTEM TURNON
label   directive   argument        comments
field     field       field           field
```

Formats are described in the following manner:

a. Any blank shown in the format requires at least one space to be inserted in the statement (optionally use more spaces).

b. All literals, characters which must be typed just as they are shown, are in upper case, e.g., EXEC.

c. All non-literals are in lower case. You must determine the proper literal to insert after reading the remainder of the format description test.

d. Optional arguments are shown encased in brackets, < >. These arguments may or may not be needed, depending on the leading arguments. Their use can only be determined by reading the text of the format description.

e. However, anything (blanks, literals, or non-literals) not encased in brackets must be included in the statement.

The examples given for each directive will help illustrate that directive's use and its options. More than one statement may be input on a line if each statement is terminated by a semicolon. The EXEC directive must be the final statement of a line if it is used in this manner.

## CRT KEYBOARD OPERATIONS

To input CCIL directives from the CRT keyboard, use the following method:

a. Press the BREAK key to stop dynamic updates of the screen.

b. Type in the required information.

c. Press the TRANSMIT key to transmit the typed-in information, and also to allow the return of dynamic updates to the screen.

Several cautions are to be observed when using the CRT keyboard. Do not press the BREAK key while a SNAP is in progress. Also, do not allow long periods of time to elapse between pressing the BREAK key and the TRANSMIT key, because a large backlog of accumulated update information which is destined for that screen can cause system operational problems. Know what you are going to type before you press the BREAK key. If, after pressing the BREAK key, you change your mind and do not wish to transmit information, clear the line and press the TRANSMIT key anyway.

# SECTION 2. CONSTANTS AND VARIABLES

# SECTION 2. CONSTANTS AND VARIABLES

## CONSTANTS

Constants are representations of data that cannot change in value. The common conception of constants, being a number rather than a symbol, is essentially correct but not sufficient for our purposes. CCIL constants may also be a character string - a symbol. This type of constant is called text. To the computer, constants are quantities that are used immediately in the expression that the machine is presently executing and then they are lost. A constant may not appear to the left of a CCIL assignment statement (refer to section 3) except as a subscript.

### Integer or Real

Again, the common concept of integer and real numbers is still a valid, but somewhat insufficient, definition. An integer is a whole quantity which has no fractional or decimal part, that is, no decimal point. A real number has a decimal point regardless of whether any numbers precede or follow the decimal point. Any variable stored in a field defined as integer will be truncated to an integer number, while variables sotred in a real field will retain their fractional parts. CCIL calls the determination of real or integer numbers the determinations of 'type.'

## Representation of Constants

| Constant Type | Representation |
|---|---|
| Hexidecimal<br>Example: X'3F7' | X'a'<br>where a = 1 to 8 hexidecimal digits, Ø-9, A-F |
| Binary<br>Example: B'1Ø11' | B'a'<br>where a = 1 to 32 binary digits, Ø or 1 |
| Octal<br>Example: O'735' | O'a'<br>where a = 1 to 11 octal digits, Ø-7 |
| Integer<br>Example: 2419 | <±>a<br>where a = decimal number less than 8 digits |
| Floating<br>Example: -2.14132 | <±>a.b<br>where *a and b = any combination of up to 14 decimal digits |
| Exponential Base 1Ø or Scientific<br>Example: 12.1E-99 | <±>a.bE<±>c<br>where *a and b = any combination of up to 14 decimal digits<br>c = decimal integer less than 1ØØ |
| Character<br>Example: C'AA' | C 'a'<br>where a = any string of characters except semicolon** |
| Text<br>Example: 'STOP' | 'a'<br>where a = any string of characters except semicolon**, a text string is not allowed in an expression |

*Either 'a' or 'b' can be missing. The decimal point is the critical item.

**Two successive apostrophes are stored as a single apostrophe.

The optional sign of a constant, <±>, when not stated, defaults to a +.

## VARIABLES

Variables, in relation to CCIL, refer to internal computer locations to which the procedure writer or the system development group has given names. These named locations can be referenced to store data; can be used in equations to generate new data to be placed in the same, or other, named locations; or can be recalled for display on the printer or CRT. If variables have been named by the systems development group then they are called global variables. If they are named by the procedure writer they are called local variables. Variables are typed as real or integer numbers similarly to constants.

### Global Variables

Global variables always exist and can be referenced or used by any procedure writer. There are three types of globals: command field definition, telemetry items, and CCIL. The latest edition of Doc. #733-76-ØØ1, IUE Telemetry and Command Users Manual, defines all global variables. All telemetry globals, and some CCIL globals must not appear to the left of the equal sign in an assignment statement, meaning the procedure writer cannot alter their value.

### Local Variables

When the procedure writer wants to create new variables to use, called local variables, the REAL and INTEGER directives may be used. Determination must be made whether the variable's use will be for whole numbers, INTEGER, or fractional or decimal numbers, REAL. If the variable's permanent use in uncertain, make it REAL. Remember, however, that any variable used for indexing, loop execution counting, or as the argument of any jump (IF, GOTO), must be defined as INTEGER.

The name chosen for a local variable must follow these rules:

1. Start with alphabetic characters.
2. Be one to eight characters in length.
3. Contain only alphabetic characters or decimal digits.

Additionally, the local variable name may have to be unique. For information on this determination and on determining the length of time a local variable will exist, refer to the following subsection on environments.

## Passed Arguments

When bringing in a procedure or subroutine with the EXEC and CALL directives, a string of up to eight arguments is allowed to be transferred. The format for each of the arguments can be any legal expression. This means that a constant, a variable, or an equation may be used. What actually gets passed is a value.

| Argument Format | Contents of Passed Agruments |
|---|---|
| Constant | Value of the constant |
| Variable Name | Value of the variable name at time of statement execution |
| Equation | Value of the equation, calculated from values of its contents at time of statement execution. |

These values are stored in arrays called AF, for procedures, and ARG, for subroutines. The arrays are accessed as AF(i) or ARG(i), 'i' being one to eight. The arrays are defined to be only as large as needed. If only three arguments on a CALL are used, then ARG(3) is the largest value referenced. An ARG(4) reference would be illegal. Writing into

an array, if defined, is permissible and does not modify any of the parameters used to generate the original value. Answers cannot be returned to the calling routine, however, as the arrays are destroyed at the RETURN statement.

## Environment

Environment refers to a set of local variables and arguments that are available to the executing code. The available set may change, however, depending on whether a procedure or a subroutine is executing and whether any internal references to other modules are made. Global variables are a part of every environment because of their universal availability. Refer to the definition of environments in table 2-1.

## Procedures

A procedure can define a set of local variables and can be called with arguments. This set, or environment, is kept unless another procedure is executed from the set or the procedure executes an exit (RETURN). The availability of the set is unaffected by subroutines; however, an EXEC directive, which transfers control to another procedure, changes the environment. If the environment changes, a new set of variables and arguments are defined for a new procedure, which is also unaffected by subroutines. There can be eight transfers of environments because nested procedures, or internal references to other modules, may be eight deep. Each time a procedure is referenced, the previous environment is stored aside (variables are saved, procedure arguments are lost) and a new environment is defined. As each procedure executes a RETURN, the current set of variables is deleted and the prior set is reinstated.

Table 2-1. Definition of Environments

| Environment | Definition |
|---|---|
| Single Statement Mode | Global variables<br>Own unique local variables |
| Procedure Mode | |
| Procedures | Global variables<br>Own unique local variables and arguments |
| Subroutines | Global variables<br>Unique local variables and arguments from calling procedure (not subroutine)<br>Own unique local variables and arguments |
| Suspended Procedure Mode | Global variables<br>Local variables and arguments defined when this mode entered |

Local variable names used in a procedure cannot be defined if they duplicate global variable names since global variables are always available. A local variable may, however, be defined for one procedure and still be unknown to any other procedures. This means local variables in one procedure may not be referenced in another procedure; yet, a local variable defined in one procedure may have the same name as a local variable defined in another procedure.

As previously stated, a procedure can define local variables and be accompanied by up to eight arguments. The value, not the names, of these arguments are stored in locations called the AF array. The array may be accessed by AF(i), 'i' being any value between one and eight. Eight arguments are available with each procedure executed within another procedure, but when this is done the previous set of arguments are lost and a new set inserted. When a RETURN is made, the most recently defined set is kept.

To summarize, all procedures retain the set of variables which were defined for the last procedure entered and the set of arguments which were defined for the deepest nested procedure entered. Duplicate names in separate procedures have no affect on the procedures. To pass a local variable from one procedure to another, it must be passed by value as an argument to the EXEC directive.

## Subroutines

Subroutines can define their local variables and have their own argument array called ARG, which is referenced by using ARG(i), 'i' being any value between one and eight. If a subroutine calls another subroutine, the previous variables are stored and the previous arguments lost. Each succeeding subroutine has no knowledge of the former

Table 2-2. Legal Local Variable Names

| Module Type | Local Variable Names |
|---|---|
| Single Statement Mode | Can be same as any name used in any procedure or subroutine |
| Procedure Mode | |
| Procedures | Can be same as any name used in Single Statement Mode. |
| | Can be same in any name used in any other procedure. |
| | Can be same as any name used in a subroutine that this procedure doesn't call. |
| | Cannot be the same as any name used in a subroutine that this procedure does call. |
| Subroutines | Can be same as any name used in Single Statement Mode. |
| | Can be same as any name used in any other subroutine. |
| | Can be same as any name used in a procedure that doesn't call this subroutine. |
| | Cannot be the same as any name used in a procedure that does call this subroutine. |

environment. As subroutines RETURN, the current environment is deleted and the previous environment reinstated (variables, not arguments), indicating that duplicate variable names in separate subroutines are never in effect concurrently, and are therefore acceptable.

A procedure must be executed to call a subroutine since a subroutine cannot be called from a keyboard. The environment of that procedure is still in effect for all subroutines referenced from within that procedure. Therefore, local variables in a subroutine cannot be defined with the same name of a local variable used in the procedure. Subroutines are able to access the AF array and local variables of the calling procedure as well as the ARG array and local variables of the subroutine.

# SECTION 3. ASSIGNMENTS

# SECTION 3. ASSIGNMENTS

## DECLARATION DIRECTIVES

All local variables (not global) must be defined as either INTEGER or REAL. INTEGER or REAL also determines whether computer space allocated to the variable is to be a single word or an array. The size of arrays should be kept as small as possible. At definition, variables are initialized to a value of zero. See page 2-1 for discussion of real and integer numbers, page 2-3 for local variable restrictions.

### INTEGER - Declare Integer Number

#### Description

The INTEGER directive defines the one or more variables that are listed as arguments, as integer numbers.

#### Format

INTEGER a(x)<,b(y),...,n(z)>

where a through n = names of local variables used in the procedure.

x through z = expressions evaluating to decimal integers $\leq$ 255, specifying the number of words to allocate for the variable. If 'x' through 'z' = 1, use as optional.

#### Example

INTEGER SUMA(1),SUMB(1),TOTSUM(1Ø) or

INTEGER SUMA,SUMB,TOTSUM(1Ø)

Defines the variables SUMA, SUMB, TOTSUM as integer numbers. SUMA and SUMB are defined as scalar variables while TOTSUM is defined as an array of length 1Ø.

## REAL - Declare Real Number

### Description

The REAL directive defines the one or more variables that are listed as arguments, as real numbers.

### Format

REAL a(x)<,b(y),...,n(z)>

where a through n = names of local variables used in the procedure.
and x through z = expressions evaluating to decimal integers ≤ 255, specifying the number of words to allocate for the variable. If 'x' through 'z' = 1, use as optional.

### Example

REAL ITEM(100),MODE(1) or
REAL ITEM(100),MODE

Defines the variables ITEM and MODE as real numbers. MODE is defined as a scalar variable while ITEM is defined as an array of length 100.

## EQV - Declare Equivalent Variable

### Description

The EQV directive is used to equate two variable names as operationally identical items. The utility of EQV is to alleviate the task of numerous statement changes in a procedure. For instance, the name of a global variable may be changed. Or perhaps two different spellings of a local variable are used. Maybe two people contributed code to a procedure, each using a different local variable name for the same physical variable. All of these local variables can be equated without changing code, as shown in the examples below.

### Format

EQV a(x)<,b(y),...,n(z)>

where a through n = variable name that is presently undefined
x through z = variable name currently defined.

Example

Suppose a routine referencing the global FRMSYC was written and then the true global name was changed to FMSYNC. To allow the code to run without further modifications, insert the following:

```
EQV FRMSYC(FMSYNC)              / PROC FRMSYC = GLOB FMSYNC
```

If there are two different names for a local variable in the procedure, say TEMP1 and TEMPA, correct the situation by declaring one or the other to be type REAL or INTEGER and then equate the two names. It doesn't matter which name is declared with the type directive. Either of the following will work.

```
REAL TEMPA
EQV  TEMP1(TEMPA)
      or
REAL TEMP1
EQV  TEMPA(TEMP1)
```

## NEWSYM - Clear Local Variable Table

Description

The NEWSYM directive clears (removes, deletes) all local variables, of the console in which it is entered, which are defined for the current level of execution. AF and ARG arrays are not effected.

Format

```
NEWSYM
```

# ASSIGNMENTS

A CCIL assignment is the transfer of a value of an expression to a variable. The transfer replaces the previous value associated with the variable. The type of the expression value (real or integer) is converted to match the variable type.

An assignment has the form:

a = b

where 'a' is a variable that will receive the calculated value of expression 'b'. Multiple assignments, or combinations of assignments and directives, may be made on a line if each statement is terminated by a semicolon. They would have the form:

REAL C;C=D;CALL SUB1Ø,C

where 'D' is again an expression whose value will be placed in the variable 'C'.

An expression is any set of operations which culminates in a single value. Among the CCIL items usable in expressions are:

- Constants
- Variables (global and local)
- Functions

These items can be used as single items or can be combined with any of the arithmetic, logical, or relational operators. The type assigned to the result of an expression depends on the type of the items used in the expression. Once a REAL type is encountered, the expression type changes to REAL, and will stay REAL, regardless of subsequent types encountered.

Operations permitted within an expression are listed in table 3-1.

Table 3-1. CCIL Operators

| Level of Precedence | Operation |
|---|---|
| 1. | Functions |
| 2. | Unary + <br> - <br> ¬ or .NOT. <br> (-) takes 2's complement of integer portion, doesn't change type <br> (¬) takes 1's complement of integer portion, doesn't change type |
| 3. | Multiplication * <br> Division / |
| 4. | Addition + <br> Subtraction - |
| 5. | Relational .EQ. or = (equal) <br> .NE. or ¬= (not equal) <br> .LT. or < (less than) <br> .GT. or > (greater than) <br> .LE. or <= (less than or equal) <br> .GE. or >= (greater than or equal) <br> All generate an integer answer; Ø=false, -1=true |
| 6. | Logical .AND. or & <br> .OR. or \| <br> Generate the AND or inclusive OR, bit by bit, of integer portions of the arguments - fractional parts of arguments will be truncated. |

Within the same level, evaluation goes left to right. Level of precedence can be overriden by use of parenthesis.

Some examples of valid expressions are:

```
A
5/2
A|X'FØ'
A.EQ.2.OR.A=3
A+3&(C+3)/2|1Ø
F(B+3*C|B'11Ø')&B'1ØØ'
FUN(R,S)¬ = ¬FUN(S,R)|1
```

## FUNCTIONS

### NUM - Find Dimension

#### Description

The NUM function evaluates the argument name and returns an integer number, which is the dimension or size of the variable used. A common use is to obtain the number of arguments passed to a procedure or subroutine using the variables AF and ARG, respectively. This value is available to that procedure or subroutine for checking against the number of arguments that are expected. If the variable is undefined, NUM returns a zero.

#### Format

NUM(a)

where a = any variable name

#### Example

```
SUBRTA SUBR                    / SUBROUTINE A REQUIRES 5 ARGS
       IF NUM(ARG).EQ.5,+3     / CHECK ARGS
       WAIT                    / INCORRECT NUMBER OF ARGS, SUBRTA
       RETURN
       CONTINUE
```

## DEVICE - Find Redundant Unit Number

### Description

The DEVICE function evaluates the argument name and returns an integer number, which is the unit number of the redundant S/C subsystem that the computer is presently addressing. Refer to the SW directive for redundant unit selection. The most common use of the unit number will be in camera subroutines where the telemetry variables are sets of arrays containing different parameters for each camera.

### Format

DEVICE(a)

where a = a subsystem name from the following list:

DMU, OBC, LVSW, FES, IRA, RW, EV, MECH, PAS, CAM.

### Example

```
UNIT=DEVICE(CAM)                / GET CAMERA UNIT NUMBER
WAIT SCANBT(UNIT).EQ.Ø          / WAIT FOR SCAN COMPLETE
        or
WAIT SCANBT(DEVICE(CAM)).EQ.Ø
```

## RAW - Find Raw Telemetry Value

### Description

The RAW function returns the unconverted telemetry count value for a given telemetry point name. Normally, telemetry point values are converted to engineering units. For status points, RAW returns Ø for 'not set' and 1 for 'set.' Converted values are Ø and -1, respectively.

### Format

RAW(a)

where a = global variable name for telemetry point

### Example

```
A=RAW(TLMVAL)/2+3
```

## ALRMHI - Find Telemetry High Alarm

### Description

The ALRMHI function returns the data base value which, when exceeded in a positive direction, generates a high alarm indication for the given telemetry point (global variable) name used as argument.

### Format

ALRMHI(a)

where a = global variable name for telemetry point

### Example

```
DISPLAY ALRMHI(TLMVAL)
```

## ALRMLO - Find Telemetry Low Alarm

### Description

The ALRMLO function returns the data base value which, when exceeded in a negative direction, generates a low alarm indication for the given telemetry point (global variable) name used as an argument.

### Format

ALRMLO(a)

where a = global variable name for telemetry point.

### Example

```
IF TLMVAL-ALRMLO(TLMVAL)<Ø.1*ALRMLO(TLMVAL),ALRMCK
                                    / CHECK STATUS IF TLM WITHIN 10% OF ALARM
```

## STATIC - Test for Static Telemetry

### Description

The STATIC function returns a true/false value that tells whether a specified telemetry value is static or dynamic. The returned value is an integer of -1 (true) if the telemetry point is static, or, Ø (false) if the telemetry point is dynamic, or updating.

Format

STATIC(a)

where a = any valid telemetry name.

Example

```
IF .NOT.STATIC(DATSYS1)&.NOT.STATIC(DATSYS2)
WAIT        /DATA SYS TLM NOT UPDATING
```

## ABS - Convert to Absolute Value

Description

The ABS function evaluates an expression for its absolute value and is typed (REAL or INTEGER) the same as that of the expression.

Format

ABS(a)

where a = any legal expression.

Example

```
NEW=ABS(A&B/C)
```

## HEX - Convert Integer to Hexadecimal

Description

The HEX function returns a character string that is the hexadecimal representation of the integer portion of the argument. The HEX function is most useful as an argument to the DISPLAY function.

Format

HEX(a)

where a = any legal expression.

Example

```
assume X=3, Y=8, Z=2.
DISPLAY HEX(X*Y/Z)              / WILL DISPLAY 'C'
```

## UNSPEC - Convert Characters to Hexadecimal

### Description

The UNSPEC function returns a character string that is the hexadecimal equivalent of the expression in floating point representation.

### Format

UNSPEC(a)

where a = any legal expression.

### Example

```
DISPLAY UNSPEC(4)                DISPLAYS 414ØØØØØ
```

## MOD - Compute Modulus

### Description

This directive returns the remainder of the quotient obtained by dividing the integer portion of expression 'a' by the integer portion of expression 'b.'

### Format

MOD(a,b)

where a = an expression used as dividend.

b = an expression used as devisor.

### Example

```
DISPLAY MOD(237.6,33.3)          / DISPLAYS 6 (237/33=7 REM 6)
```

## BIT - Converts Bits to Decimal

### Description

The BIT function evaluates a string of expressions each of which evaluates to a value from 1 through 32, indicating a bit to be set. Bits are numbered from right to left in the word. Up to 32 expressions may be used as arguments to the function.

Format

BIT(a<,b,...,z>)

where a,...,z = any legal expression.

Example

```
A=BIT(5,1)                    / CREATES WORD WITH BITS 1 AND 5 SET.
```

## PPRNUM - Find Data Base Record PPR Number

Description

The PPRNUM function returns an integer corresponding to the PPR number of a named data base record.

Format

PPRNUM(a<,b>)

where a = name assigned to the PPR

b = name assigned to the PPR group - if 'b' is not present, the telemetry group (DFTTLM) is assumed.

Example

```
DISPLAY PPRNUM(DATSYS1)          / DISPLAYS 1347
DISPLAY PPRNUM(FES1,DFESPHYS)    / DISPLAYS 7211
```

# SECTION 4. CONTROL DIRECTIVES

# SECTION 4. CONTROL DIRECTIVES

## ALL MODES

### Comments

#### Description

Comments are any nonexecutable text, that is, information to the operator rather than directives to the computer. The CCIL format allows comments to be placed on the end of an executable statement. The comment field must be preceded by a minimum of one blank space to terminate the previous field. A slash (/) character starts the comment by informing the computer that the remaining data is nonexecutable. Data within the comment field, the text, may use any alphanumeric or special character except the semicolon (;). The semicolon serves as a statement terminator, so any data after the semicolon will be interpreted as a new statement.

If there is no executable statement on the input, whether input is a card or from the CRT keyboard, the slash (/) may be placed in any column from 1 to 72. Additionally, if the total input is to be a comment, an asterisk (*) may be used, but in column one only. Since CRT input effectively starts at column 2, the type of comment is illegal from the CRT keyboards. Blank comment cards, with an asterisk in column one, can be used as spacers when writing a procedure.

Either type of comment is printed and displayed in the same manner as an executable statement. Comments are counted when determining the destination of the ±n type of jumps (IF and GOTO).

#### Example

```
       WAIT              / COMMENT AT END OF EXECUTABLE STATEMENT
        / COMMENT WITHOUT EXECUTABLE STATEMENT
/ COMMENT STARTING IN COLUMN ONE
*
* COMMENT STARTING IN COLUMN ONE
*
```

## EXEC - Start Procedure Execution

### Description

The EXEC directive causes execution of a pre-defined set of statements specifically structured to be a procedure, which is stored on the current procedures file. Up to eight arguments can be passed to the procedure. The procedure may use and modify these arguments, referencing them as AF(i), but their existence is deleted on return to the calling environment. A procedure may contain an EXEC function of its own, and may process up to eight internal referenced EXEC functions in a procedure. That is, a procedure may execute a second procedure, which in turn executes a third procedure. Reference should be made to the preceding discussion on procedures for information on the handling of arguments in nested procedures.

### Format

EXEC y<,$a_1$,$a_2$,...,$a_8$>

where y = a label (procedure name) that exists on the procedure file that has a PROC as its directive.

each $a_1$,...,$a_8$ = a constant, variable, or expression passed as an argument to the procedure.

### Example

EXEC SYSAON,5,TIME,3.2,VECT(3)

Requests a procedure named SYSAON to be run. Passes 4 arguments AF(1)=5, AF(2)=TIME, AF(3)=3.2, AF(4)=VECT(3). The SYSAON procedure will reference these arguments as AF(i).

EXEC SYSAOFF

Requests a procedure named SYSAOFF to be run. No arguments are passed. Therefore, AF(i) is undefined for all values of 'i.'

EXEC TEST1,C'INIT',3,A-B*C,CLOCK,VECT(3),D&C,A>1

Requests a procedure named TEST1 to be run. The argument list shows some of the various forms that can be used: alphanumeric characters, constants, mathematical expressions, single value variables, dimensioned variables, logical expressions, and relational expressions.

## STEP - Select Procedure Execution Speed

### Description

The STEP directive controls the speed of execution of a procedure. In single-step, a hold is effectively inserted between each procedure statement. A GO statement is then required to be input from the keyboard to execute each statement. The value of STEP is passed from mode to mode as part of the environment. That is, establishing single-step in single-statement mode will cause single-step to be in effect when a procedure is executed out of single-statement mode. A second procedure executed from the first will still be in single-step. If the second procedure changes to continuous-execution, this condition will remain in effect until control is passed back to procedure one. Then the condition reverts to single-step. The same explanation holds for a subroutine called from procedure-one.

### Format

STEP a

where a = an expression which evaluates to Ø or non-zero.

= Ø
for continuous-execution or,

= non-zero
for single-step.

Example

STEP 1

Selects single-step mode. Speed of procedure execution dependent on operator's typing of GO in between each procedure statement.

STEP Ø

Selects continuous-execution mode. Speed of procedure execution dependent only on computer processing time.

## PROCEDURE ONLY

### PROC - Define Procedure Entry Point

Description

The PROC directive defines an allowable procedure entry point (label) that may be referenced by an EXEC function. Procedures can have only one entry point.

Format

PROC

Example

See combined example following PEND directive.

### SUBR - Define Subroutine Entry Point

Description

The SUBR directive defines an allowable subroutine entry point (label) that may be referenced by a CALL function. Subroutines can have only one entry point.

Format

SUBR

Example

See combined example following PEND directive.

## PEND - Define Procedure or Subroutine Physical Termination Point

### Description

The PEND directive defines the physical end (last card) of the card deck used to generate a procedure or subroutine. If executed, the PEND directive functions as a RETURN directive.

### Format

PEND

### Example

```
TESTA     PROC                  / START TESTA
           .
           .
           .
          CALL MODA              / CALL SUBROUTINE
           .
           .
           .
          RETURN                 / EXIT FROM PROCEDURE
          PEND                   / TERMINATE PROCEDURE DECK
MODA      SUBR                   / START MODA
           .
           .
           .
          RETURN                 / EXIST FROM SUBROUTINE
          PEND                   / TERMINATE SUBROUTINE DECK
```

## RETURN - Return From Subroutine or Procedure

### Description

The RETURN directive causes the subroutine or procedure to return control to the calling unit. Each subroutine and procedure should contain at least one RETURN directive. Multiple RETURN's may be used. The current environment is deleted and the calling environment is restored.

### Format

RETURN

### Example

See combined example following PEND directive.

## CALL - Start Subroutine Execution

### Description

The CALL directive causes execution of a pre-defined set of statements, specifically structured to be a subroutine, which is stored on the current procedures file. Up to eight arguments can be passed to the subroutine. The subroutine may use and modify these arguments, referencing them as ARG(i), but their existence is deleted on return to the calling routine. You may process up to eight "nested" CALL functions. That is, a subroutine may call a second subroutine, which in turn calls a third subroutine. Reference should be made to the preceding discussion of subroutines for information on the handling of arguments in nested subroutines.

### Format

CALL y<,$a_1$,$a_2$,...,$a_8$>

where y = a label (subroutine name) that exists on the procedure file that has a SUBR as its directive.

each $a_1$,...,$a_8$ = a constant, variable, or expression passed as an argument to the subroutine.

### Example

CALL CAM1X1,TIME,NUMB(4),15

Requests a subroutine named CAM1X1 to be run. Passes three arguments ARG(1)=TIME, ARG(2)=NUMB(4), ARG(3)=15.

The CAM1X1 subroutine will reference these arguments as ARG(i).

CALL CAM1OF

Requests a subroutine named CAM1OF to be run. No arguments are passed.

Therefore, ARG(i) is undefined for all values of 'i.'

## CONTINUE - Do Nothing

### Description

The CONTINUE directive is used primarily as a place holder in procedures. It is also commonly used as the cycle instruction of a DO loop. An implied CONTINUE, or do-nothing, directive is formed by constructing a statement without any directive. This might be a statement with only a label, or a label and a comment, or simply a comment.

### Format

```
      CONTINUE
                                  / COMMENT
LABEL
LABEL                             / COMMENT
```

### Example

See combined example following the DO directive.

## GOTO - Jump

### Description

The GOTO directive causes the execution of the procedure or subroutine to jump from its present position to a new specified position. You may only jump to positions that are still within the body of the currently executing procedure or subroutine. The GOTO directive cannot be used as the cycle instruction of a DO loop.

### Format

GOTO a

where a = either:

- = the name of a label in the procedure or subroutine to which the execution should be transferred, or;
- = an expression evaluating to ±n, n being an integer number of statements (see note below) to skip.

Example

```
      GOTO PARTB                  / JUMP TO PARTB
       :
       :
      (MISC PROCEDURE)
       :
       :
PARTB CONTINUE
      GOTO -5                     / JUMP BACKWARDS 5 STATEMENTS
```

Note: GOTO's using the ±n argument should be used with caution. The number of statements should be kept small so that their range can easily be seen and modified when editing procedures. The number of statements skipped will include comment cards. Remember that a multiple assign card counts as multiple statements.

## IF - Test Conditions

### Description

The IF directive provides the capability of testing the value of selected variables for the purpose of decision making. It can be used, for example, for limit testing, for determining options within a procedure, or for testing external conditions relating to procedure operation.

The expression is evaluated and assigned the value 'true' (-1) or 'false' (zero). If the expression is true, the procedure will jump to a new location determined by the final argument field. The new location can be either a label, plus or minus a specified number of statements, or, if there is no argument following the expression, plus two statements (a skip of one statement). If the expression is false, the procedure will continue with the next sequential statement. Jumps using the ±n argument need to be used with the same caution as GOTO6.

Refer to the note in the GOTO-Jump procedure. In either case, a jump to a label or a jump of ±n statements, the jump must remain within the body of the currently executing module.

Format

IF a<,b>

where a = logical expression, simple or compound.

b = procedure destination if expression 'a' is true, in one of the following forms:

= blank - skip one statement

= an expression evaluating to ±n - jump forward or backwards n statements

= label - jump to statement having this label.

Example

```
     IF IMODE.EQ.4,MODE4          / IF IMODE=4,GO TO LABEL MODE4,
*ELSE GO TO NEXT SEQ STATEMENT
     IF A.LT.2.1.OR.A.GT.2.9,NOGOOD    / IF A IS EITHER LESS THAN 2.1 OR
*GREATER THAN 2.9, GO TO LABEL NOGOOD.  ELSE GO TO NEXT SEQ STATEMENT.
     IF NEW.NE.OLD,+3             / IF NEW IS DIFFERENT FROM OLD, JUMP 3
*STATEMENT, ELSE GO TO NEXT SEQ STATEMENT.
     IF A.EQ.B.AND.A.EQ.C         / IF A=B=C SKIP ONE STATEMENT, ELSE GO
*TO NEXT SEQ STATEMENT.
```

## WAIT - Wait For Conditions

Description

The WAIT directive causes a temporary or unconditional suspension of the module until the conditions of the WAIT are met. There are four forms of WAIT, one for unconditional, one for specified time, and two for a condition coupled with a maximum time, one of which also having a cycle rate specified for testing the condition. If the maximum time to wait for a condition is exceeded, the system notifies the operator and enters a HOLD state.

Format

WAIT

which is an unconditional wait, really a HOLD, and must be answered with a GO

or

WAIT a

where a = an expression evaluating to a decimal constant specifying number of seconds to wait before proceeding

or

WAIT x,y

where x = an expression (condition) to be evaluated as true (non-zero) before proceeding

y = an expression evaluating to a decimal constant specifying number of seconds, a value used as the maximum time to wait regardless of whether condition 'x' is true.

or

WAIT x,y,z

where x and y = same as the format for WAIT x,y.

z = period of time, in seconds, between evaluations of 'x.' If 'z' is not specified, default is to a Ø. One-second evaluation cycle.

Example

```
WAIT 5.5                          / WAIT 5.5 SECONDS
WAIT SC28V.GT.26,12Ø              / WAIT FOR 28V ON OR 12Ø SECONDS
WAIT SC28V.GT.26,12Ø,15.Ø         / SAME AS ABOVE, EXCEPT EVALUATION
*CYCLE IS 15 SECONDS INSTEAD OF Ø.1 SECONDS.
WAIT                              / END TEST, CHECK S/C
```

## DO - Start Loop Execution

Description

The DO directive instructs the procedure to execute a specified sequence of statements in a cycle for a specified number of times. The sequence

begins with the statement following the DO statement and ends with the statement on which a particular label is defined. This 'label statement' must physically follow the DO statement. Historically, the 'label statement' has also been a CONTINUE directive for compatibility with PCL. The actual limitations for CCIL are that a DO loop must not end with a 'GOTO' action, that being either a GOTO or an IF directive. Nesting DO loops, that is, a DO loop within a DO loop, is allowed. The nested DO loop must be totally contained within the previous loop, however. See the example below. Code in a procedure, which branches into a loop, is acceptable. Note, though, that since the DO directive will not have been executed, the execution of the statement containing the termination label will have no meaning and the procedure will never loop, but will always continue in sequential operations.

A review of the following format paragraph will clarify the remainder of this paragraph. One item of interest is the value of the loop counter 'i' at loop completion. As stated in the following paragraph, loop completion is made when the count variable 'i' exceeds the value of the terminating variable 'n'. The specific action at the label statement is this: Increment 'i' by 'p'; test 'i' greater than 'm'; loop if not greater, exit if greater. Obviously, since the test is made at the end of the loop, not at the beginning, the sequence will always be performed once. This also means that the DO loop ending with label SUB1Ø2 will execute three times, but 'i' will have a value of four at loop completion (refer to the example in the following paragraph). A related fact is that the 'i' in the loop may be modified. Loop exiting will still be accomplished whenever 'i' exceeds 'm'. One may also branch out of a loop without waiting for the prescribed number of loop cycles. The loop counter 'i' will continue to have whatever value it had when the branch was made. Finally, while 'i' may be modified, 'm', 'n', or 'p' should not be changed within the loop.

Format

DO a i=m,n<,p>

where a = name of label which terminates the loop, which must not prefice a GOTO or IF directive.

i = name of an integer variable used to count loop cycles.

m = an expression which evaluates to a positive or negative integer to which 'i' is initially set.

n = an expression which evaluates to a positive or negative integer specifying a value of 'i' which, when exceeded, will terminate the cycling.

p = optionally, an expression which evaluates to a positive integer whose value is added to 'i' each pass through label 'a'. If 'p' is not specified, 'i' will increment by 1.

Example

```
SUB1Ø1   CONTINUE                 / USED AS PLACE HOLDER
         DO SUB1Ø2  I=1,3         / CYCLE 3 TIMES
         :EV                      / PULSE JETS
         WAIT 30                  / WAIT 30 SECONDS
         SNAP ENGVAL              / DOCUMENT
SUB1Ø2   CONTINUE
         JEN=Ø                    / DISABLE
*TWO EXAMPLES OF LEGAL NESTED LOOPS
         DO SUB2Ø5  I=-6,Ø,2      / CYCLE 3 TIMES
             :
         DO SUB2Ø4  I=-2,Ø        / CYCLE 2 TIMES
             :
SUB2Ø4   CONTINUE
SUB2Ø5   CONTINUE
```

```
        DO SUB21Ø  I=-6,Ø,2    / CYCLE 3 TIMES
            :
        DO SUB21Ø  I=-2,Ø      / CYCLE 2 TIMES
            :
SUB21Ø  CONTINUE
```

## HOLD - Suspend Procedure Execution

### Description

The HOLD directive is normally issued from the keyboard during procedure execution and its effect is to suspend the execution of the active procedure. If no argument is used, the procedure will halt at conclusion of the processing of the present statement. If an argument is specified, the procedure will halt when that statement of the procedure is accessed, but before execution of the statement. The HOLD 'argument' condition is referenced to the current environment and will be deleted on exiting a module.

### Format

HOLD <a>

where a = any valid label or statement number (including comment).

### Example

Refer to the combined example following the GO directive.

## PROCEDURE AND SUSPENDED PROCEDURE

### ABORT - Terminate Procedure Execution

#### Description

The ABORT directive unconditionally terminates the execution of an active procedure and returns system to the single-statement, or keyboard, mode. If the ABORT is requested from the keyboard and an action has been initiated by the procedure, that action will be completed normally before procedure termination, with the exception of the WAIT directive. Any WAIT directive will be cancelled immediately by the ABORT. The optional character string is not processed any differently than the normal comment field in CCIL.

Note that ABORT is not the equivalent of RETURN. RETURN will cause an exit from the present environment and a restoration of the previous environment. By the nesting of EXEC and CALL directives, the procedure may be several stages deep in previous environments. ABORT exits all procedure-type environments and returns to single-statement mode.

#### Format

ABORT <'optional character string'>

### GLBABORT - Terminate All Procedures

#### Description

The GLBABORT directive is the equivalent of ABORT except it acts on a system, rather than single-keyboard, scale. Any procedures, executing from any console, will be terminated.

#### Format

GLBABORT

## SUSPENDED PROCEDURE ONLY

### GO - Resume Procedure Execution

#### Description

The GO directive resumes execution of a suspended procedure. If no argument is used, procedure will resume at next sequential statement. If an argument is specified, the procedure will jump to the start of the statement that is specified and then resume execution. The requested statement must reside within the body of the currently executing module. This function is legal from keyboard only.

#### Format

GO <a>

where a = any valid label or statement number within the body of present module. Only one argument is allowed. Statement number can be a comment record.

#### Example

Suppose a portion of a procedure looked this this:

```
112   TESTØ6    ISA=511
113             ILA=511
114             SSR=Ø
115             LSR=Ø
116   CMDØ22    :SCAN
(rest of procedure)
538   TESTØ7    ISA=1Ø23
539             ILA=1Ø23
540             SSR=1Ø23
541             LSR=1Ø23
542             WAIT / OPTIONAL CMD USED - DETERMINE PROPER ACTION
```

As the procedure was running, suppose a command should not be executed at statement 116 with the options given it in the proceeding statements 112-115. Instead the options in statements 538-541 should be given.

From the keyboard, before statement 116 is reached, issue either of the functionally equivalent statements:

```
HOLD   116
or
HOLD   CMDØ22
```

The procedure will then be suspended as statement 116 is accessed, but before its execution. Now, from the keyboard, issue either of the functionally equivalent statements:

```
GO   538
or
GO   TESTØ7
```

The procedure will resume execution by jumping to statement 538. The new options will be set and the procedure will pause (stop) at statement 542. Now, from the keyboard, issue either of the functionally equivalent statements:

```
GO   116
or
GO   CMDØ22
```

The procedure resumes execution by jumping to statement 116, issues the scan command, and continues through the procedure.

# SECTION 5. COMMAND DIRECTIVES

# SECTION 5. COMMAND DIRECTIVES

Commanding is done by transferring a 60-bit string of information to the spacecraft. These 60 bits consist of segments of data which are set by CCIL directives. For serial commands there are three primary fields.

First, there are 8 bits which specify the command decoder address. The spacecraft receiver directs the command to one of the two command decoders, based on this value. The value is set by CCIL using the SET DECODER directive.

Secondly, there are 6 bits which specify the command address. Each piece of spacecraft hardware has a unique address. This address is set by CCIL using the SW directive. For example, issuing SW FES,1 will cause the number 39 (serial command address number) to be placed in the command field (if a :FES is issued). When the spacecraft command decoder decodes the data, it will send the information to FES1 (39) rather than FES2 (47) or any other piece of hardware.

Finally, there is a 37-bit field which is built by combining the values of one or more global command variables assigned to each type of command.

Sending a command, by using the :command directive (assume :FES), causes the computer to transfer the contents of the variables for FES global variables, the FES SW setting, and the SET DECODER setting to a command buffer, along with some other information. The resulting 60 bits is then transmitted to the spacecraft. This generation is shown pictorially in figure 5-1.

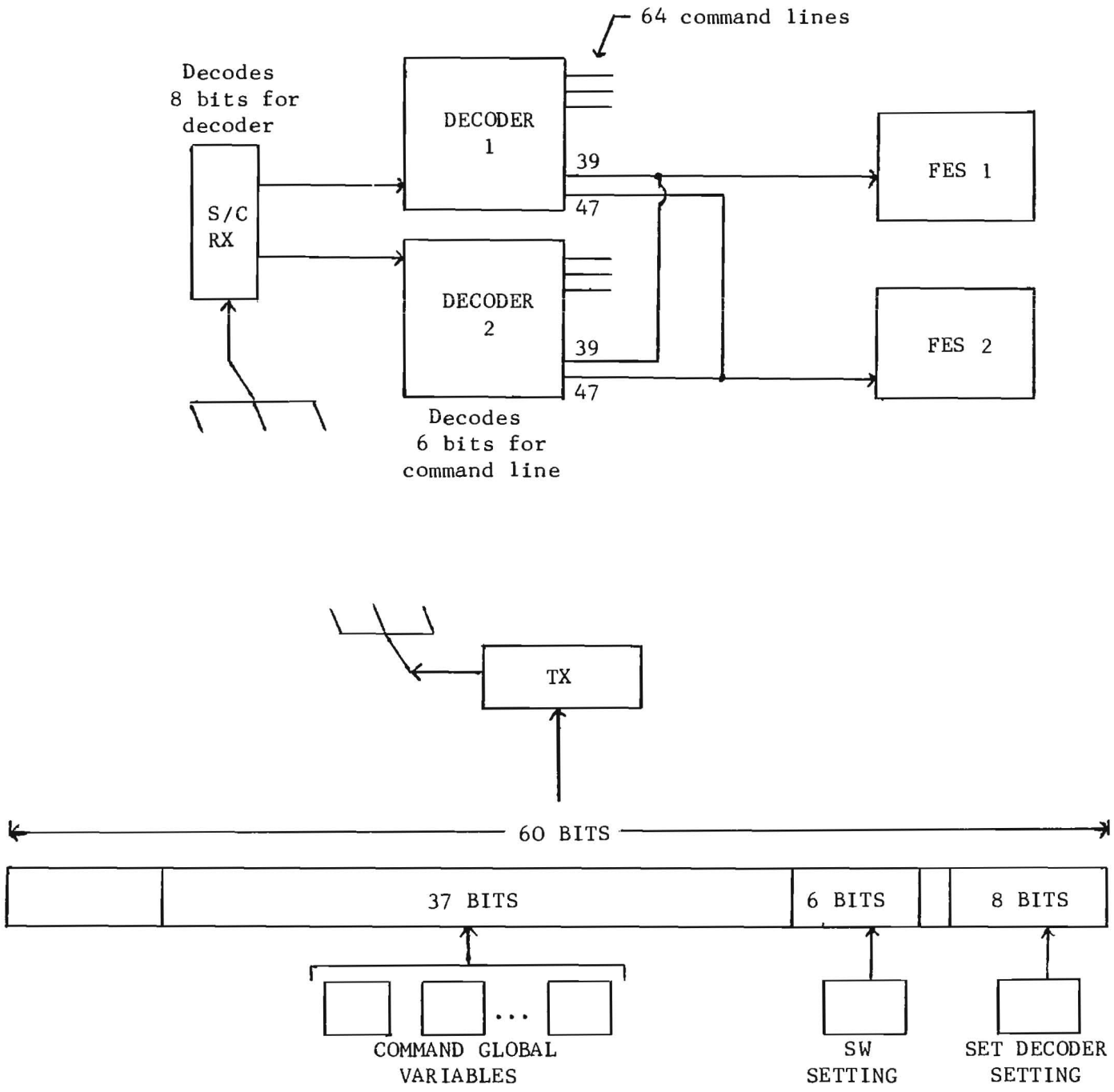



Figure 5-1. Cor nd Generation

Setting the variables associated with commands is accomplished in several ways. The most universal way is to use the assign routine (Variable=x) to set proper values into the command variable before actually calling the command. A second way is to set the variables in the same statement as the issuance of the command. A third method of commanding is to follow, on the same line (in the same record, CCIL), with a string of arguments.

Specific commands use one or more of the above methods. The various methods of commanding, and the groups of commands using these methods, are illustrated following the discussion of command handling directives.

Once set, the variables are constant and do not have to be reset. The variables should be initialized in each procedure. Either individually assign a value to each one or execute a procedure that you know assigns values to all command variables. The EXAMINE SERIAL directive will display these values. Do not issue commands without knowing what is in all variables which define the total impact of that command. Although it does not ensure proper command fabrication, command variables are checked against an internal table of allowable values at the time a command transmission is requested.

Capabilities have been provided for creating and executing a buffer of commands (a command sequence) so that time-critical commanding can be performed. There will be two CRT pages which will display the contents of this buffer.

## SET DECODER - Format Commands for Redundant Encoder Selection (CCIL Unique)

### Description

The SET DECODER directive is used to direct the construction, and therefore the transmission of commands to one of the two spacecraft command decoders. Specifically, it directs the computer to format all subsequent commands with a particular number in the decoder address field (bits 1 to 8) of the command. The SET DECODER directive should not be issued while commands are being uplinked to the spacecraft.

### Format

SET DECODER,a

where a = an expression that evaluates to 1 or 2;
meaning the selection of redundant command decoder 1 or 2.

### Example

```
SET DECODER,1                    / FORMAT CMDS FOR DECODER 1
```

## SW - Format Commands for Redundant Unit Selection

### Description

The SW directive is used to direct the construction, and therefore the transmission of commands to a particular piece of redundant spacecraft hardware. More precisely, part of its function is to specify the ground formatting of a particular portion of the command field. This field, the serial command address (bits 1Ø to 15), is eventually used by the IUE spacecraft's command decoder to direct the command to the proper redundant device. SW also directs the computer to use a particular group

of command variables in the generation of the serial data for the command. Finally, SW directs the computer to have all further assignments of data to non-subscripted command variables to be assigned to the group associated with the redundant unit selected. Values may still be assigned to nonselected units by subscripting the variable.

Format

SW a,n

where a = any one of the following subsystems:

| | |
|---|---|
| DMU | (Data Multiplexer Unit) |
| OBC | (On-Board Computer) |
| LVSW | (Low Voltage Switch) |
| FES | (Fine Error Sensor) |
| IRA | (Inertial Reference Assembly) |
| RW | (Reaction Wheel) |
| EV | (Engine/Valve) |
| MECH | (EEA Mechanism) |
| PAS | (Panoramic Scanner) |
| CAM | (Camera) |

and where n = an integer expression equating to 1 or 2;
meaning the selection of either redundant unit 1 or 2, except for subsystem CAM, where-
n = an integer expression equating to 1, 2, 3, or 4;
meaning the selection of one of the 4 redundant units.

Example

```
INTEGER  UNIT
UNIT=2
SW DMU,1                    / SWITCH TO DMU 1
SRATE=1                     / SRATE(1)=1, 4Økb
SRATE(2)=2                  / SRATE(2)=2, 2Økb
:DMU                        / USING ARRAY(1) AND SERIAL COMMAND
                            / ADDRESS 6, BUILD AND TRANSMIT CMD, 4Økb
SW DMU,UNIT                 / SWITCH TO DMU 2
SRATE=Ø                     / SRATE(2)=Ø, 8ØKB
:DMU                        / USING ARRAY(2) AND SERIAL COMMAND
                            / ADDRESS 14, BUILD AND TRANSMIT CMD
```

## EXAMINE SERIAL - Display Command Parameters (CCIL Unique)

### Description

The EXAMINE SERIAL directive displays the SW setting, being the redundant device selected, and all command variables associated with the requested command, including those for non-selected redundant units. The information is output to the event printer and to the requesting CRT. If a reference is going to be to the CRT, the DBASEB page should be displayed prior to issuing the EXAMINE SERIAL directive. The system doesn't stop to allow a review of the data, so a WAIT should be inserted to verify the configuration.

### Format

EXAMINE SERIAL,a

where a = any one of the following:

- DMU (Data Multiplexer Unit)
- EV (Engine/Valve)
- FES (Fine Error Sensor)
- MECH (EEA Focus)
- IRA (Internal Reference Assembly)

PRECESS (Precession and Nutation)
PAS (Panoramic Scanner)
RW (Reaction Wheel)
SCAN (SSCL Scan)
SIHTR (SSCL Camera Setup)
SIALGN (SSCL Camera Setup)
SIUVC (SSCL Camera Setup)
SIMODE (SSCL Camera Mode)

Example

```
EXAMINE SERIAL,DMU              / DISPLAY DMU CMD PARAMETERS
```

## SET TRANSMIT - Select Command Transmission Mode (CCIL Unique)

Description

The SET TRANSMIT directive specifies the command transmission mode.

Format

```
SET TRANSMIT,a
where a = either:
        = SINGLE
          for transmit (and verify) commands one at a time, or;
        = MULTIPLE
          for transmit (and verify) commands normally.
```

Example

```
SET TRANSMIT,SINGLE             / TRANSMIT COMMANDS ONE AT A TIME
```

## SET CMDMODE - Select Command Ground-Transmission Path (CCIL Unique)

Description

The SET CMDMODE directive selects the ground path taken for command transmissions.

Format

SET CMDMODE,a

where a = either:

= CMDENC

for command transmission through the command encoder, meaning direct link to the transmitter, or;

= DDPS

for command transmission through the NASA link to a remote site (SCE).

Example

```
SET CMDMODE,CMDENC                / SET UP FOR DIRECT COMMANDING
```

## SET VERIFY - Select Verification Mode (CCIL Unique)

Description

The SET VERIFY directive specifies the type of command transmission verification to be performed.

Format

SET VERIFY,a<,b>

where a = OFF

for no verification is performed, or;

where a,b = one or both of:

= OCC

for verification is performed by control center software, or;

= SCE

for verification is performed by Spacecraft Command Encoder (SCE) site software.

OCC and SCE verification may be requested together. The OCC verification has precedence over the SCE verification. The format of this request is as follows:

SET VERIFY,OCC,SCE

## REPLY CMD - Process Command Requests (CCIL Unique)

### Description

The REPLY CMD directive is used to respond to one of two types of messages from the commanding portion of CCIL. One, before a command is transmitted, it is compared with the table of critical commands. If the desired command is critical, the following message is displayed.

CRITICAL - message - description (nn)
where message = reason for command being critical.
description = command description.
nn = command position in command display.

Secondly, if a command transmission fails, one of the following messages is displayed.

TELECOMMUNICATIONS RESPONSE TIMEOUT
COMMAND UPLINK FAILURE
VERIFICATION FAILURE
TELEMETRY TIMEOUT DURING COMMANDING - CA
COUNTER UNSTABLE PRIOR TO COMMANDING - CB
COUNTER OUTSIDE OF LIMITS - CD
VERIFICATION FAILURE - CE
NO TELEMETRY AT START OF COMMANDING - CF
DECODER OFF - CH
DECODER UNSTABLE PRIOR TO COMMANDING - CI

In any of these cases, no further command processing is done until the REPLY CMD directive is issued.

Format

REPLY CMD,a

where a = either:

= SKIP

for skip transmission or retransmission of this command.

= CLEAR

for skip transmission or retransmission of this command and clear entire command buffer (not necessarily just command displayed).

= APPROVE

for transmission of critical command.

= RETRANS

for retransmission of failed command.

Example

```
REPLY CMD,CLEAR                 / CLEAR ALL CMDS IN CMD BUFFER
```

## NORMAL SERIAL COMMANDS

The following set of commands can be sent with their respective variables set either:

1. before the command is executed, that, is on separate lines either singly or in multiple assigns, or

2. on the same line (record, CCIL) as the command, either singly or in multiple assigns with each assign separated by a comma.

Note that in method (1) the multiple assigns are separated by semicolons, not commas. While this is one line for PCL, it is multiple records for CCIL. Some examples follow:

Setting single variables before command:

```
XVAM=Ø                          / ENABLE TM VAM
:DMU                            / TRANSMIT
ISA=128
ILA=128
SSR=128
LSR=128                         / SET PARAMETERS
:SCAN                           / START SCAN
```

Setting multiple variables before command:

```
ISA=128;ILA=128;SSR=128;LSR=128
:SCAN
```

Setting variable on line with command:

```
:DMU XVAM=Ø                     / ENABLE TM VAM
:SCAN ISA=128,ILA=128,SSR=128,LSR=128
```

In addition to the three totally distinct methods shown, the many combinations of the three methods are all equally legal. Such as:

```
ISA=128
ILA=128;SSR=128
:SCAN LSR=128
```

## :DMU - Data Multiplexer Unit

### Description

Commands the Data Multiplexer Unit (DMU). Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW DMU,a directive. The values of the two variable

sets and the SW setting are displayed by issuing the EXAMINE SERIAL,DMU directive. Variables used to generate the :DMU command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| IA1 | Indirect Address, Number 1 | Ø/15 |
| IA2 | Indirect Address, Number 2 | Ø/15 |
| XVAM | Ø = Cycle T/M VAM and Load OBC VAM | Ø/1 |
| | 1 = Cycle OBC VAM and load T/M VAM | |
| XCLK | Ø = Redundant Clock | Ø/1 |
| | 1 = Main Clock | |
| AROM | Ø = Single Address Format | Ø |
| | 1 = Alternate Format (not used on IUE) | |
| CFMT | Ø = Direct Computer Format | Ø/3 |
| | 1 = NOP | |
| | 2 = OBC VAM Computer Format | |
| | 3 = ROM Computer Format | |
| TFMT | Ø = VAM Telemetry Format | Ø/1 |
| | 1 = ROM Telemetry Format | |
| TMROM | Ø = 1A Transfer Orbit | Ø/3 |
| | 1 = 2A Mission | |
| | 2 = 1B SI Video | |
| | 3 = 2B OBC Memory | |
| CODED | Ø = Block Code (uncoded) | Ø/1 |
| | 1 = Convolutional (coded) | |
| MXR | Multiplex Ratio: | Ø/6 |
| | Ø = All Telemetry | |
| | 1 = 1:1 | |
| | 2 = 2:1 | |
| | 3 = 4:1 | |
| | 4 = 8:1 | |
| | 5 = 16:1 | |
| | 6 = 32:1 | |

| NAME | USE | MIN/MAX |
|---|---|---|
| SRATE | Transfer Ratio:<br>Ø = 8ØKB<br>1 = 4ØKB<br>2 = 2ØKB<br>3 = 1ØKB<br>4 = 5KB<br>5 = 2.5KB | Ø/5 |

:EV - Engine/Valve

Description

Commands the engine and valves. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW EV,a directive. The values of the two variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,EV directive. Variables used to generate the :EV command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| LPUL | Low Thrust Engine Mode:<br>Ø = Continuous<br>1 = Pulse | Ø/1 |
| PHASE | Accelerometer Phase:<br>Ø = Non-Invert<br>1 = Invert | Ø/1 |
| ACC | Accelerometer Select:<br>Ø = Accelerometer 1<br>1 = Accelerometer 2 | Ø/1 |
| EVC | Mode Control:<br>Ø = Secondary Mode Control<br>1 = Primary Mode Control | Ø/1 |
| HPUL | High Thrust Engine Mode<br>Ø = Continuous<br>1 = Pulse | Ø/1 |

| NAME | USE | MIN/MAX |
|---|---|---|
| VALVE | One bit per valve* with the MSB being valve 1 and LSB being valve 7:<br>Ø = Closed Valve<br>1 = Open Valve | Ø/127 |
| ENG | One bit per engine** with the MSB being engine 1 and the LSB being engine 12:<br>Ø = Disable Engine<br>1 = Enable Engine | Ø/4Ø95 |
| EVE | E/V Enable:<br>Ø = Disable bits 1 to 24 of this command<br>1 = Enable bits 1 to 24 of this command | Ø/1 |
| FIRE | One bit per engine** with engine 1 being the MSB and engine 12 being the LSB:<br>Ø = Engine Shutdown<br>1 = Engine Fire | Ø/4Ø95 |

*VALVE can be set with constants V1 through V7 for valves 1 through 7.

**ENG and FIRE can be set with constats J1 through J12 for iets 1 through 12.

Example:

```
VALVE = V2+V4+V7              / OPEN VALVES 2,4,AND 7
ENG = J1+J12;FIRE = J1+J12  / ENABLE AND FIRE ENGINES 1 AND 12
```

## :FES - Fine Error Sensor

### Description

Commands the Fine Error Sensor. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW FES,a directive. The values of the two variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,FES directive. Variables used to generate the :FES command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| FESX | Frame start coordinate coarse offset | Ø/127 |
| FESY | Line start coordinate coarse offset | Ø/127 |
| FESL | Frame and Line Length | Ø/63 |
| FLAP | Underlap:<br>Ø = Overlap<br>1 = Underlap | Ø/1 |
| FESXF | Horizontal Fine Positioning frame | Ø/31 |
| FESYF | Vertical Fine Positioning Line | Ø/31 |
| FESTE | Track Enable:<br>Ø = Map Only<br>1 = Map then track | Ø/1 |
| FESTSR | Track Scan Rate Change<br>Ø = Fast<br>1 = Slow | Ø/1 |
| FESTHD | Threshold:<br>Ø = +11<br>1 = +1Ø<br>2 = +9<br>3 = +8 | Ø/3 |

| NAME | USE | MIN/MAX |
|---|---|---|
| FESSM | System Mode:<br>Ø = Primary<br>1 = Search and Track<br>2 = Field Camera | Ø/2 |

:FOCUS - EEA Focus

Description

Command the EEA Focus. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW MECH,a directive. The values of the two variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,MECH directive. The variable used to generate the :FOCUS command is:

| NAME | USE | MIN/MAX |
|---|---|---|
| FOC | Focus Drive Select:<br>1 = PHI1A<br>2 = PHI2A<br>4 = PHI1B<br>8 = PHI2B | 1/8 |

:IRA - Inertial Reference Assembly

Description

Commands the Inertial Reference Assembly. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW IRA,a directive. The values of the two variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,IRA directive. Variables used to generate the :IRA command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| | Mode Control:* | |
| IRAMC | One bit per mode control MSB being control 1 and LSB being control 6;<br>Ø = Rate<br>1 = Hold/Slew | Ø/63 |

| NAME | USE | MIN/MAX |
|---|---|---|
| | Gyro Select:* | |
| IRAGYR | One bit per gyro MSB being gyro 1 and LSB being gyro 6;<br>Ø = OFF<br>1 = ON | Ø/63 |
| | Heater Select:* | |
| IRAHTR | One bit per heater MSB being heater 1 and LSB being heater 6;<br>Ø = LO (4.5 watts)<br>1 = HI (12 watts) | Ø/63 |
| IRACOLD | Rate Cold:<br>Ø = Rate Normal<br>1 = Rate Cold | Ø/1 |
| IRAQB | Qualifier Bit:<br>Ø = Disable control of IRAMC, IRAGYR, IRAHTR, and IRACOLD<br>1 = Enable control of IRAMC, IRAGYR, IRAHTR, and IRACOLD | Ø/1 |
| IRABD | Bias Data Bit | Ø/1Ø23 |
| IRABA | Bias Address:<br>B'ØØØØ' = Disable<br>B'ØØØ1' = Pitch Active<br>B'ØØ1Ø' = Roll Active<br>B'Ø1ØØ' = Yaw Active | Ø/7 |

*IRAMC,IRAGYR, and IRAHTR can be set with contstants GY1 through GY6.

Example:

```
IRAGYR = GY3+GY6              / SELECT GYROS 3 AND 6
```

## :NUTAT - Nutation

### Description

Commands nutation. There is no redundant unit. The value of the variable set is displayed by issuing the EXAMINE SERIAL,PRECESS directive. Variables used to generate the :NUTAT command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| ZERADJ | Zero crossing adjust<br>-.14 + ZERADJ*.017V | Ø/15 |
| TGA | Gain Adjust:<br>Ø = -Ø.52V+TADJ*Ø.Ø147V<br>1 = -Ø.72V+TADJ*Ø.Ø147V | Ø/1 |
| NUTIN | Ø = Nutation OFF<br>1 = Nutation ON | Ø/1 |
| NUTEN | Nutation Register Update<br>(always 1 for nutation) | 1/1 |
| TADJ | Threshold Adjustment<br>Ø.Ø14V/Count | Ø/15 |

## :PRECESS - Precession

### Description

Commands precession. There is no redundant unit. The value of the variable set is displayed by issuing the EXAMINE SERIAL,PRECESS directive. Variables used to generate the :PRECESS command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| PSUN | Sun Pulse:<br>Ø = Select SMSS Pulse<br>1 = Select PAS Pulse | Ø/1 |
| PLONG | Sector Fire Count | Ø/15 |
| PESUN | Fire Pulse:<br>Ø = Sun Disable<br>1 = Fire on Sun Pulse | Ø/1 |
| PSTART | Start Sector | Ø/127 |
| PFIRE | Number of Precession Firings | Ø/255 |

## :PAS - Panoramic Scanner

### Description

Commands the panoramic scanner. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW PAS,a directive. The values of the two variable sets and of the SW setting is displayed by issuing the EXAMINE SERIAL,PAS directive. Variables used to generate the :PAS command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| PASMOD | Ø = Spherical Mode | Ø/1 |
| | 1 = Planar Mode | |
| PASCLK | Ø = 125ØHz clock, 78.ØHz step | Ø/3 |
| | 1 = 625Hz clock, 39.ØHz step | |
| | 2 = 312Hz clock, 19.5Hz step | |
| | 3 = 156Hz clock, 9.8Hz step | |
| PASDIR | Ø = CCW Scan Direction | Ø/1 |
| | 1 = CW Scan Direction | |
| PASMAX | Theta Maximum Angle | Ø.Ø/36Ø.Ø |
| PASLEW | Ø = Slew Disable | Ø/1 |
| | 1 = Slew Enable | |
| PASCAN | Ø = Select Continuous Submode Scan | Ø/1 |
| | 1 = Select Sector Scan | |
| PASSUN | Ø = Select PAS Sun Sensor | Ø/1 |
| | 1 = Select SMSS Sun Sensor | |
| PASMIN | Theta Minimum Angle | Ø.Ø/36Ø.Ø |

## :RW - Reaction Wheel

### Description

Commands the reaction wheels. Selection of one of two redundant sets of variables and command paths is accomplished by issuing the SW RW,a directive. The values of the two variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,RW directive. Variables used to generate the :RW command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| RWMODE | Mode Decoder, defined as follows: | Ø/15 |
| | Symbols used in following definitions. | |

| Symbol | Definition | |
|---|---|---|
| R1 | Enable Rate 1 to C&M Card | |
| R2 | Enable Rate 2 to C&M Card | |
| CSS | Enable Coarse Sun Sensor | |
| S3 | C&M CMD to Wheel Driver | |
| S1 | Enable D/A 1 to Wheel Driver | |
| S2 | Enable D/A 2 to Wheel Driver | |
| | Reset | Ø |
| R1,CSS | Sun Acquisition Jet-IRA Rate 1 (Gyro 1,3,5) | 1 |
| R1,CSS,S3 | Sun Acquisition Wheel-IRA Rate 1 (Gyro 1,3,5) | 2 |
| R2,CSS | Sun Acquisition Jets-IRA Rate 2 (Gyro 2,4,6) | 3 |
| R2,CSS,S3 | Sun Acquisition Wheel-IRA Rate 2 (Gyro 2,4,6) | 4 |
| R1 | Jet Rate Damp-IRA Rate 1 (Gyro 1,3,5) | 5 |
| R1,S3 | Wheel Hold-IRA Rate 1 (Gyro 1,3,5) | 6 |
| R2 | Jet Rate Damp-IRA Rate 2 (Gyro 2,4,6) | 7 |
| | Reset | 8 |
| R2,S3 | Wheel Hold-IRA Rate 2 (Gyro 2,4,6) | 9 |
| CSS,S3 | Wheel Sunbath-Powder Restore Mode | 10 |
| S1 | Dighold, Slew, Mode, Gradcntrl, Opnloop | 11 |
| S2 | Dighold, Slew, Mode, Grndcntrl, Opnloop | 12 |

| NAME | USE | MIN/MAX |
|---|---|---|
| | Reset | 13 |
| | Reset | 14 |
| | Reset | 15 |
| RWENAB | Reaction Wheel Mode Control:<br>Ø = Disable<br>1 = Enable | Ø/1 |
| REDUN | Redundant Wheel Command | Ø/255 |
| ROLL | Roll Wheel Command | Ø/255 |
| YAW | Yaw Wheel Command | Ø/255 |
| PITCH | Pitch Wheel Command | Ø/255 |

## :SCAN - SSCL Scan

### Description

Commands the SSCL scanner. Selection of one of four redundant sets of variables and command paths is accomplished by issuing the SW CAM,a directive. The values of the four variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,SCAN directive. Variables used to generate the :SCAN command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| ISA | Starting Sample (Initial Sample Address-895) | Ø/1Ø23 |
| ILA | Starting Line (Initial Line Address-895) | Ø/1Ø23 |
| LSR | Number of Lines (Line Scan Range-768) | Ø/1Ø23 |
| SSR | Number of Samples per Line (Sample Scan Range-768) | Ø/1Ø23 |

## :SIHTR - SSCL Camera Setup

### Description

Executes setup of SSCL camera. Selection of one of four redundant sets of variables and command paths is accomplished by issuing the SW CAM,a directive. The values of the four variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,SIHTR directive. Variables used to generate the :SIHTR command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| HTR | Heater Voltage | Ø/127 |
| SEC | Secondary High Voltage Gain Setting | Ø/127 |
| G1 | Grid 1 DAC Setting | Ø/127 |

In addition to setting the variables (NAME) equal to a value, they may be set equal to the following predefined constants:

| NAME | CONSTANT | DEFINITION |
|---|---|---|
| G3 | G3D | Grid 3 Defocused |
| | G3F | Grid 3 Focused |
| YAL | YAN | Y-Alignment Normal |
| | YAMC | Y-Alignment Minimal Cntrd |
| XAL | XAN | X-Alignment Normal |
| | XAMC | X-Alignment Minimal Cntrd |

## :SIALGN - SSCL Camera Setup

### Description

Executes setup of SSCL camera. Selection of one of four redundant sets of variables and command paths is accomplished by issuing the SW CAM,a directive. The values of the four variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,SIALGN directive. Variables used to generate the :SIALGN command are:

| NAME | USE | MIN/MAX |
|---|---|---|
| G3 | Grid 3 DAC Setting | Ø/127 |
| YAL | Y-Alignment DAC | Ø/127 |
| XAL | X-Alignment DAC | Ø/127 |

In addition to setting the variables (NAME) equal to a value, they may be set equal to the following predefined constants:

| NAME | CONSTANT | DEFINITION |
|---|---|---|
| HTR | HTL | Heater Voltage Low |
| | HTH | Heater Voltage High |
| SEC | SEC1 | Max Gain (Expose) |
| | SEC2 | Med Gain (Expose) |
| | SEC3 | Min Gain (Expose) |
| | SEC4 | Prep Gain |
| | SECMX | Max SEC |
| G1 | G1CO | Grid 1 Cutoff |
| | G1ED | Grid 1 Erase Defocused |
| | G1EF | Grid 1 Erase Focused |
| | G1RD | Grid 1 Read |
| | G1MX | Grid 1 Voltage |

## :SIUVC - SSCL Camera Setup

### Description

Executes setup of SSCL camera. Selection of one of four redundant sets of variables and command paths is accomplished by issuing the SW CAM,a directive. The values of the four variable sets and of the SW setting are displayed by issuing the EXAMINE SERIAL,SIUVC directive. The variable used to generate the :SIUVC command is:

| NAME | USE | MIN/MAX |
|---|---|---|
| UVC | Ultraviolet Converter Digital to Analog Gain Setting | Ø/127 |

In addition to setting UVC equal to a value, it may be set equal to the following predefined constants:

| CONSTANT | DEFINITION |
|---|---|
| UVC1 | Max Gain (Expose) |
| UVC2 | Med Gain (Expose) |
| UVC3 | Min Gain (Expose) |
| UVC4 | Prep |
| UVCOF | UVC Cutoff |
| UVCMX | Max UVC |

## SPECIAL SERIAL COMMANDS

The following set of commands are exceptions to the rule of setting variables by way of assign statements. These commands have an argument string which must follow the command on the same line.

### :APER - EEA Aperture

Description

Requests a command to select the EEA aperture position. Selection of one of two redundant sets of command paths is accomplished by issuing the SW MECH,a directive.

Format

:APER <a>

where no argument = deselect aperture, or

where a = OPEN
for aperture open, or;
= CLOSE
for aperture closed.

Example

```
:APER OPEN                    / OPEN EEA APERTURE
```

## :CAMSEL - EEA Camera Select

### Description

Requests a command to EEA mechanism camera select. Selection of one of two redundant sets of command paths is accomplished by issuing the SW MECH,a directive.

### Format

:CAMSEL <a>

where no argument = deselect all cameras, or

where a = SWR
for redundant short wavelength;
= SWP
for prime short wavelength;
= LWR
for redundant long wavelength;
= LWP
for prime long wavelength.

### Example

```
:CAMSEL SWP                    / SELECT PRIME SHORT WAVELENGTH CAMERA
```

## :DISP - EEA Dispersion

### Description

Requests a command to select EEA dispersion. Selection of one of two redundant sets of command paths is accomplished by issuing the SW MECH,a directive.

Format

:DISP <a>

where no argument = deselect dispersion, or

where a = SWL
for short wavelength low;
= SWH
for short wavelength high;
= LWL
for long wavelength low;
= LWH
for long wavelength high.

Example

:DISP LWL / SELECT LONG WAVELENGTH LOW

## :SIMODE - SSCL Mode

Description

Requests a command to SSCL camera for mode determination. Selection of one of four redundant command paths is accomplished by issuing the SW CAM,a directive. The value of the four variable sets and of the SW setting is displayed by issuing the EXAMINE,SERIAL,SIMODE directive. The variable used to generate the "SIMODE command is:

| NAME | USE | MIN/MAX |
|---|---|---|
| SZ | STEP SIZE | Ø/3 |

SZ needed to be assigned before command transmission.

Format

:SIMODE <a,...,n>

where no argument = clear camera mode, or

where a through n = zero, one or more of the following parameters:

| Parameter | Use |
|---|---|
| WLC | Wavelength Calibration Lamp |
| TF2 | Tungsten Flood Lamp Enable PS # 2 |
| FB | Fudicial and Backhole Lamp Enable |
| TF1 | Tungsten Flood Lamp Enable PS # 1 |
| UVF | UV Flood Lamp Enable |

and zero, or one and only one, of the following:

| | |
|---|---|
| EX | Expose |
| ER | Erase Fast |
| RDER | Erase Read Rate |
| RDLO | Read Lo Gain |
| RDHI | Read Hi Gain |
| STBY | Standby |

Example

```
:SIMODE FB,RDHI                 / ENABLE FB LAMP, READ HIGH GAIN
```

## :CRU - Command Relay Unit

Description

Requests a command to the command relay unit. Multiple relays actuation will be sent as a sequence of commands, one command for each relay specified. There is no redundant command path for the :CRU command.

Format

:CRU a,m<,n,...,z>

where a = ON
for setting relays on;
= OFF
for setting relays off.

m - z = integer constants from 1 to 64 representing relays to be set. Table 5-1 contains the listing of relays and their function.

Example

```
:CRU ON,1,6,1Ø                    / SET LVPS1,DMU,OBC2 RELAYS ON
```

## :LVSW - Low Voltage

Description

Controls the application of low voltage to the S/C subsystems. Any combination of the 1Ø relays can be controlled with a single command. Selection of one of two redundant command paths is accomplished by issuing the SW LVSW,a directive.

Format

:LVSW <a,m,b,n,...,g,x>

where no arguments = all devices off, or

where a - g = one or more of the subsystem names,

| | | | | |
|---|---|---|---|---|
| MUX1 | MEC1 | SWR | LWR | FES1 |
| MUX2 | MEC2 | SWP | LWP | FES2 |

m - x = either:
= ON or;
= OFF

Example

```
:LVSW MUX1,ON,MEC1,ON,LWP,ON     / TURN ON CAMERA 1
:LVSW                            /TURN OFF ALL DEVICES
```

## Table 5-1. U Command Listing

```
          (13)     (14)              (10)
          +-------+----------------+---------+
          |NOTUSED|CONTROL(ON/OFF)|RELAY I.D|
          +-------+----------------+---------+
           1                               37
```

| CMD | SYS | FUNCTION | TELEMETRY STATUS | *ON | OFF | **ID |
|---|---|---|---|---|---|---|
| 1 | EEA | POWER SUPPLY NO.1 | ASC2-57 <0.1V=OFF >4.0V=ON | 27 | 26 | 37 |
| 2 | EEA | SEC. MIRROR HTR CIRCUIT NO.1 | NDT | 27 | 26 | 36 |
| 3 | ACS | ENGINE/VALVE DRIVER NO.1 | SB-22 0=OFF 1=ON | 27 | 26 | 35 |
| 4 | ACS | WDA POWER SUPPLY NO.1 | ASC2-0 <0.1V=OFF >3V=ON | 27 | 26 | 34 |
| 5 | ACS | IRA GYRO NO.1 | DSC-0/1 BIT 21 0=OFF 1=ON | 27 | 26 | 33 |
| 6 | DMU | DATA SYSTEM NO.1 | SB-8 0=OFF 1=ON | 27 | 26 | 32 |
| 7 | SBAND | TRANSMITTER POWER RELAY A | ASC1-2/3 <0.1V=OFF>3.5V=ON | 27 | 26 | 31 |
| 8 | SI | MODE SELECT NO.2 | NDT | 27 | 26 | 30 |
| 9 | SI | SUN SHUTTER ELECTRONICS | NDT | 27 | 26 | 29 |
| 10 | OBC | OBC NO.2 POWER | ASC2-55 <.1V=OFF >4.5V=ON | 27 | 26 | 28 |
| 11 | SI | CAL LAMP PWR SUPPLY NO.1 | DSC-30 0=OFF 1=ON | 25 | 24 | 37 |
| 12 | EEA | POWER SUPPLY NO.2 | ASC2-59 0.0V=OFF 4.5V=ON | 25 | 24 | 36 |
| 13 | SI | MODE SELECT NO.1 | NDT | 25 | 24 | 35 |
| 14 | ACS | ENGINE/VALVE DRIVER NO.2 | SB-23 0=OFF 1=ON | 25 | 24 | 34 |
| 15 | ACS | WDA POWER SUPPLY NO.2 | ASC2-5 <0.1V=OFF >3V=ON | 25 | 24 | 33 |
| 16 | ACS | IRA GYRO NO.2 | DSC-0/1 BIT 22 0=OFF 1=ON | 25 | 24 | 32 |
| 17 | DMU | DATA SYSTEM NO.2 | SB-19 0=OFF 1=ON | 25 | 24 | 31 |
| 18 | SI | FOCUS DRIVE NO.1 | NDT | 25 | 24 | 30 |
| 19 | SI | SCEM NO.4 SHORT WAVE REDUNDANT | DSC-27 0=OFF 1=ON | 25 | 24 | 29 |
| 20 | SI | SUN SHUTTER DIRECT DRIVE | NDT | 25 | 24 | 28 |
| 21 | ACS | FINE DIGITAL SUN SENSOR NO.1 FSS | ASC1-59 <0.1V=OFF >3V=ON | 23 | 22 | 37 |
| 22 | SI | CAL LAMP PWR SUPPLY NO.2 | DSC-31 0=OFF 1=ON | 23 | 22 | 36 |
| 23 | SI | PRI. MIRROR HTR CIRCUIT NO.1 | NDT | 23 | 22 | 35 |
| 24 | SI | SCEM NO.1 LONG WAVE PRIME | DSC-24 0=OFF 1=ON | 23 | 22 | 34 |
| 25 | SBAND | POWER AMP NO.1 | SB-9 0=OFF 1=ON | 23 | 22 | 33 |
| 26 | ACS | PITCH WHEEL DRIVER +28V | ASC2-7 <24V=ON | 23 | 22 | 32 |
| 27 | ACS | IRA GYRO NO.3 | DSC-0/1 BIT 23 0=OFF 1=ON | 23 | 22 | 31 |
| 28 | ACS | IRA COMMON ELECTRONICS NO. 1 | DSC-0 BIT 13 0=OFF 1=ON | 23 | 22 | 30 |
| 29 | SI | FOCUS DRIVE NO.2 | NDT | 23 | 22 | 29 |
| 30 | SI | SUN SHUTTER DIRECTION SEL. | NDT | 23 | 22 | 28 |
| 31 | ACS | PAS NO.1 | DSC-28 0=OFF 1=ON | 21 | 20 | 37 |
| 32 | ACS | IRA COMMON ELECTRONICS NO. 2 | DSC-1 BIT 14 0=OFF 1=ON | 21 | 20 | 36 |
| 33 | SI | FOCUS POSITION ELECTRONICS | NDT | 21 | 20 | 35 |
| 34 | SI | PRI. MIRROR HTR CIRCUIT NO.2 | NDT | 21 | 20 | 34 |
| 35 | SI | SCEM NO.2 LONG WAVE REDUNDANT | DSC-25 0=OFF 1=ON | 21 | 20 | 33 |
| 36 | SBAND | POWER AMP NO.2 | SB-10 0=OFF 1=ON | 21 | 20 | 32 |
| 37 | ACS | YAW WHEEL DRIVER +28V | ASC2-8 <24V=ON | 21 | 20 | 31 |
| 38 | ACS | IRA GYRO NO.4 | DSC-0/1 BIT 24 0=OFF 1=ON | 21 | 20 | 30 |
| 39 | OBC | OBC NO.1 CONFIGURATION PWR. | ASC2-3 <.1V=OFF >4.5V=ON | 21 | 20 | 29 |
| 40 | SI | FOCUS LAUNCH HOLD | NDT | 21 | 20 | 28 |
| 41 | SI | APERTURE SELECT NO.1 | NDT | 19 | 18 | 37 |
| 42 | SBAND | POWER AMP NO.3 | SB-11 0=OFF 1=ON | 19 | 18 | 36 |
| 43 | HAPS | ARM HEATER GROUP NO.1 | NDT | 19 | 18 | 35 |
| 44 | ACS | IRA GYRO NO.5 | DSC-0/1 BIT 25 0=OFF 1=ON | 19 | 18 | 34 |
| 45 | ACS | PAS NO.2 | DSC-29 0=OFF 1=ON | 19 | 18 | 33 |
| 46 | SI | CAMERA SELECT NO.1&DECK HEATER NO.1 | NDT | 19 | 18 | 32 |
| 47 | SI | SEC. MIRROR HTR CIRCUIT NO.2 | NDT | 19 | 18 | 31 |
| 48 | ACS | ROLL WHEEL DRIVER +28V | ASC2-9 <24V=ON | 19 | 18 | 30 |

Table 5-1. CRU Command Listing (Continued)

| CMD | SYS | FUNCTION | TELEMETRY STATUS | *ON | OFF | **ID |
|---|---|---|---|---|---|---|
| 49 | SBAND | TRANSMITTER 1/2 SELECT | ASC1-2 XMTR1 >3.5V=ON<br>ASC1-3 XMTR2 >3.5V=ON | 19 | 18 | 29 |
| 50 | OBC | OBC NO.2 CONFIGURATION PWR. | ASC2-55 <.1V=OFF >4.5V=ON | 19 | 18 | 28 |
| 51 | ACS | REDUNDANT WHEEL DRIVER +28V | ASC2-10 <24V=ON | 17 | 16 | 37 |
| 52 | SI | APERTURE SELECT NO.2 | NDT | 17 | 16 | 36 |
| 53 | ACS | FINE DIGITAL SUN SENSOR NO.2 FSS | ASC1-60 <0.1V=OFF >3V=ON | 17 | 16 | 35 |
| 54 | ACS | SPIN MODE SUN SENSOR-SMSS | ASC2-1 <0.1V=OFF >3V=ON | 17 | 16 | 34 |
| 55 | SI | CAMERA SELECT NO.2&DECK HEATER NO.2 | NDT | 17 | 16 | 33 |
| 56 | SBAND | TRANSMITTER POWER RELAY B | ASC1-2/3 <0.1V=OFF>3.5V=ON | 17 | 16 | 32 |
| 57 | SBAND | POWER AMP NO.4 | SB-12 0=OFF 1=ON | 17 | 16 | 31 |
| 58 | SI | SCEM NO.3 SHORT WAVE PRIME | DSC-26 0=OFF 1=ON | 17 | 16 | 30 |
| 59 | OBC | OBC NO.1 POWER | ASC2-3 <0.1V=OFF >4.5V=ON | 17 | 16 | 29 |
| 60 | ACS | IRA GYRO NO.6 | DSC-0/1 BIT 26 0=OFF 1=ON | 17 | 16 | 28 |
| 61 | PYRO | SOLAR ARRAY & SI COVER ARM1 | DSC-23 0=DISARM 1=ARM | 15 | 14 | 37 |
| 62 | PYRO | APOGEE BOOST MOTOR ARM1 | DSC-22 0=DISARM 1=ARM | 15 | 14 | 36 |
| 63 | PYRO | APOGEE BOOST MOTOR ARM2 | DSC-22 0=DISARM 1=ARM | 15 | 14 | 35 |
| 64 | PYRO | SOLAR ARRAY & SI COVER ARM2 | DSC-23 0=DISARM 1=ARM | 15 | 14 | 34 |

NOTE:
DSC=DIGITAL SUB COM
ASC=ANALOG SUB COM
SB=STATUS BIT
NDT=NO DIRECT TELEMETRY

BIT STRUCTURE:
BITS 1 THRU 13 NOT USED
BITS 14 THRU 27 ARE CONTROL BITS
BITS 28 THRU 37 ARE RELAY I.D.BITS
* ON AND OFF REFER TO CONTROL BIT.
** ID REFERES TO RELAY ID FIELD.

SERIAL COMMAND 8 HEX ADDRESS 88

*Reprinted from RCA "Telemetry and Command Manual," (IUE-733-76-101).

## OTHER COMMANDS

### COMMAND - Build Unique Command

#### Description

Provides the capability of specifying any 37-bit configuration as a serial command. The COMMAND function generates a serial command and stores it in a buffer. A :SEND function is needed for execution.

#### Format

COMMAND $a,x_1,y_1<,...,x_n,y_n>$

where a = the serial address (Ø to 63) of the desired command,

x(i) = an expression which evaluates to an integer number defining the number of bits to be used (field size) in representing y(i)-the sum of all x(i)'s must equal 37, any individual 'x' can range from 1 to 32.

y(i) = an expression which evaluate to an integer number specifying the value to be assigned to x(i). The right-most 'x' number of bits of 'y' will be used. Negative values of 'y' are treated as 2's complement numbers.

#### Example

```
COMMAND 14,7,X'7F',1Ø,X'3FF',2Ø,X'FFFFF'
```

Specifies a command for address 14 with every bit set to 1.

```
1st 7 bits    = X'7F'    = 111 1111
next 1Ø bits  = X'3FF'   = 11 1111 1111
final 2Ø bits = X'FFFFF' = 1111 1111 1111 1111 1111
```

### :SEND - Execute Unique Command

#### Description

Requests that the serial command built by a COMMAND function be sent for a specified number of times, at a specified interval. If optional arguments are missing, command is sent one time. The data is not destroyed.

Format

:SEND <i,n>

where i = an expression which evaluates to a decimal number specifying the interval, in seconds between commands (minimum of Ø.1);

n = an expression which evaluates to an integer number specifying the total number of commands to send.

Example

```
COMMAND 14,7,X'7F',3Ø,X'3FFFFFFF'     / BUILD COMMAND
:SEND 1.5,2Ø                          / SEND AT 1.5 SEC RATE, 2Ø TIMES
:SEND                                 / SEND 1 TIME
```

## :IMP - Execute Impulse Command

Description

Requests a series of one or more impulse commands. Multiple impulse command requests will be sent as a sequence of commands. Repeated commands will be repeated. A request for a series of impulse commands will be transmitted faster than requests for each command separately.

Format

:IMP a<,b,...,n>

where a - n = positive integer constants ≤127 representing specific impulse commands. Table 5-2 contains the impulse commands.

Example

```
:IMP 3,17,1Ø7
```

Table 5-2. [illegible]mpulse Commands

| CMD# | SYS | FUNCTION | VERIFICATION | HEX |
|---|---|---|---|---|
| 0 | | NOT USED | N/A | 000 |
| 1 | PWR | BOOST REGULATOR NO.1 ON | DSC CH.12 BIT 8=0 | 100 |
| 2 | PWR | BOOST REGULATOR NO.1 OFF | DSC CH.12 BIT 8=1 | 080 |
| 3 | PWR | BOOST REGULATOR NO.2 ON | DSC CH.13 BIT 8=0 | 180 |
| 4 | PWR | BOOST REGULATOR NO.2 OFF | DSC CH.13 BIT 8=1 | 040 |
| 5 | PWR | CHARGE REGULATOR NO.1 ON | DSC CH.12 BIT 6=0 | 140 |
| 6 | PWR | CHARGE REGULATOR NO.1 TRIKLE CH | DSC CH.12 BIT 6=1 | 0C0 |
| 7 | PWR | BATTERY NO.1 TRICKLE CHARGE HI | DSC CH.12 BIT 5=0 | 1C0 |
| 8 | PWR | BATTERY NO.1 TRICKLE CHARGE LO | DSC CH.12 BIT 5=1 | 020 |
| 9 | PWR | CHARGE REGULATOR NO.2 ON | DSC CH.13 BIT 6=0 | 120 |
| 10 | PWR | CHARGE REGULATOR NO.2 TRIKLE CH | DSC CH.13 BIT 6=1 | 0A0 |
| 11 | PWR | BATTERY NO.2 TRICKLE CHARGE HI | DSC CH.13 BIT 5=0 | 1A0 |
| 12 | PWR | BATTERY NO.2 TRICKLE CHARGE LO | DSC CH.13 BIT 5=1 | 060 |
| 13 | PWR | BATTERY&3RD ELECTRODE NO.1 ON | DSC CH.12 BIT 7=0 & BIT 1=0 | 160 |
| 14 | PWR | BATTERY&3RD ELECTRODE NO.1 OFF | DSC CH.12 BIT 7=1 & BIT 1=1 | 0E0 |
| 15 | PWR | BATTERY&3RD ELECTRODE NO.2 ON | DSC CH.13 BIT 7=0 & BIT 2=0 | 1E0 |
| 16 | PWR | BATTERY&3RD ELECTRODE NO.2 OFF | DSC CH.13 BIT 7=1 & BIT 2=1 | 010 |
| 17 | | SPARE | | 110 |
| 18 | PWR | 3RD ELECTRODE NO.1 OFF | DSC CH.12 BIT 1=1 NOTE 2 | 090 |
| 19 | OBC | OBC 2 CPU OFF | DSC CH.10 | 190 |
| 20 | PWR | 3RD ELECTRODE NO.2 OFF | DSC CH.13 BIT 2=1 NOTE 2 | 050 |
| 21 | PWR | BATTERY NO.1 UV DETECTOR ON | DSC CH.12 BIT 4=0 | 150 |
| 22 | PWR | BATTERY NO.1 UV DETECTOR OFF | DSC CH.12 BIT 4=1 | 0D0 |
| 23 | PWR | BATTERY NO.2 UV DETECTOR ON | DSC CH.13 BIT 4=0 | 1D0 |
| 24 | PWR | BATTERY NO.2 UV DETECTOR OFF | DSC CH.13 BIT 4=1 | 030 |
| 25 | PWR | +28 BUS UV DETECTOR ON | DSC CH.12 BIT 2=0 | 130 |
| 26 | PWR | +28 BUS UV DETECTOR OFF | DSC CH.12 BIT 2=1 | 0B0 |
| 27 | PWR | +28 BUS OC DETECTOR ON | DSC CH.12 BIT 3=0 | 1B0 |
| 28 | PWR | +28 BUS OC DETECTOR OFF | DSC CH.12 BIT 3=1 | 070 |
| 29 | PWR | AUTOMATIC LOAD REMOVE DISABLE | DSC CH.13 BIT 3=1 NOTE 3 | 170 |
| 30 | PWR | AUTOMATIC LOAD REMOVE ENABLE | DSC CH.13 BIT 3=0 | 0F0 |
| 31 | RF | VHF NO.1 TRANSMITTER ON | ASC1 CH.4>3.8V | 1F0 |
| 32 | RF | VHF NO.1 TRANSMITTER OFF | ASC1 CH.4<0.1V | 008 |
| 33 | RF | VHF NO.1 RANGING ON | OBSERVE RANGING DATA | 108 |
| 34 | RF | VHF NO.1 RANGING OFF | RANGING DATA LOSS | 088 |
| 35 | RF | NO.1 MOD SOURCE NO.1 (DMU 1) | OBSERVE TM WITH DMU 1 ON | 188 |
| 36 | RF | NO.1 MOD SOURCE NO.2 (DMU 2) | OBSERVE TM WITH DMU 2 ON | 048 |
| 37 | RF | VHF NO.2 TRANSMITTER ON | ASC1 CH.5>3.8V | 148 |
| 38 | RF | VHF NO.2 TRANSMITTER OFF | ASC1 CH.5<0.1V | 0C8 |
| 39 | RF | VHF NO.2 RANGING ON | OBSERVE RANGING DATA | 1C8 |
| 40 | RF | VHF NO.2 RANGING OFF | RANGING DATA LOSS | 028 |
| 41 | RF | NO.2 MOD SOURCE NO.1 (DMU 1) | OBSERVE TM WITH DMU 1 ON | 128 |
| 42 | RF | NO.2 MOD SOURCE NO.2 (DMU 2) | OBSERVE TM WITH DMU 2 ON | 0A8 |
| 43 | SI | EEA LOW VOLTAGE SW-SELECT P.S.1 | NDT | 1A8 |
| 44 | SI | EEA LOW VOLTAGE SW-SELECT P.S.2 | NDT | 068 |
| 45 | SI | SUN SHUTTER OPEN | STATUS BITS 7=0,6=0 | 168 |
| 46 | SI | SUN SHUTTER CLOSE | STATUS BITS 6=1,7=1 | 0E8 |
| 47 | SI | EEA LOW VOLTAGE SW.-ALL LOADS OFF | NDT | 1E8 |
| 48 | CMD | COMMAND DECODER OFF (NOTE 1) | NO.1=SB 20/NO.2=SB 21 =0 | 018 |
| 49 | OBC | OBC NO.1 CPU ON | DSC CH.9 | 118 |
| 50 | OBC | OBC NO.1 CPU OFF | DSC CH.9 | 098 |
| 51 | OBC | OBC NO.2 CPU ON | DSC CH.10 | 198 |
| 52 | OBC | OBC 1 UPPER MEMORY BUS ON | DSC CH.18 = 1 (FF) | 058 |
| 53 | OBC | OBC 1 UPPER MEMORY BUS OFF | DSC CH.18 = 0 (00) | 158 |
| 54 | OBC | OBC 1 LOWER MEMORY BUS ON | DSC CH.19 = 1 (FF) | 0D8 |
| 55 | OBC | OBC 1 LOWER MEMORY BUS OFF | DSC CH.19 = 0 (00) | 1D8 |
| 56 | OBC | OBC 2 UPPER MEMORY BUS ON | DSC CH.20 = 1 (FF) | 038 |
| 57 | OBC | OBC 2 UPPER MEMORY BUS OFF | DSC CH.20 = 0 (00) | 138 |

Table 5-2. Impulse Commands (Continued)

| CMD# | SYS | FUNCTION | VERIFICATION | HEX |
|---|---|---|---|---|
| 58 | OBC | OBC 2 LOWER MEMORY BUS ON | DSC CH.21 = 1 (FF) | 088 |
| 59 | OBC | OBC 2 LOWER MEMORY BUS OFF | DSC CH.21 = 0 (00) | 188 |
| 60 | SI | FOCUS ELECTRONICS 1-START FOCUS | EXPERIMENT DATA VERIFY | 078 |
| 61 | SI | FOCUS ELECTRONICS 2-START FOCUS | EXPERIMENT DATA VERIFY | 178 |
| 62 | SI | FINE ERROR SENSOR 1-SEARCH ADVANCE | EXPERIMENT DATA VERIFY | 0F8 |
| 63 | SI | FINE ERROR SENSOR 2-SEARCH ADVANCE | EXPERIMENT DATA VERIFY | 1F8 |
| 64 | | SPARE | | 004 |
| 65 | ACS | PITCH WHEEL DRIVER-CONVERTER 1 | NO DIRECT VERIFY | 104 |
| 66 | ACS | PITCH WHEEL DRIVER-CONVERTER 2 | NO DIRECT VERIFY | 084 |
| 67 | ACS | YAW WHEEL DRIVER-CONVERTER 1 | NO DIRECT VERIFY | 184 |
| 68 | ACS | YAW WHEEL DRIVER-CONVERTER 2 | NO DIRECT VERIFY | 044 |
| 69 | ACS | ROLL WHEEL DRIVER-CONVERTER 1 | NO DIRECT VERIFY | 144 |
| 70 | ACS | ROLL WHEEL DRIVER-CONVERTER 2 | NO DIRECT VERIFY | 0C4 |
| 71 | ACS | REDUNDANT WHEEL DRIVER-CONVERTER 1 | NO DIRECT VERIFY | 1C4 |
| 72 | ACS | REDUNDANT WHEEL DRIVER-CONVERTER 2 | NO DIRECT VERIFY | 024 |
| 73 | ACS | PRECESSION/NUTATION-CONVERTER 1 | NO DIRECT VERIFY | 124 |
| 74 | ACS | PRECESSION/NUTATION-CONVERTER 2 | NO DIRECT VERIFY | 0A4 |
| 75 | ACS | COMPENSATION/MIXING-CONVERTER 1 | | 1A4 |
| 76 | ACS | COMPENSATION/MIXING-CONVERTER 2 | | 064 |
| 77 | ACS | WHEEL CMD D/A 1-CONVERTER 1 | NO DIRECT VERIFY | 164 |
| 78 | ACS | WHEEL CMD D/A 1-CONVERTER 2 | NO DIRECT VERIFY | 0E4 |
| 79 | ACS | WHEEL CMD D/A 2-CONVERTER 1 | NO DIRECT VERIFY | 1E4 |
| 80 | ACS | WHEEL CMD D/A 2-CONVERTER 2 | NO DIRECT VERIFY | 014 |
| 81 | ACS | ACCELEROMETER A-CONVERTER 1 | NO DIRECT VERIFY | 114 |
| 82 | ACS | ACCELEROMETER A-CONVERTER 2 | NO DIRECT VERIFY | 094 |
| 83 | ACS | ACCELEROMETER B-CONVERTER 1 | NO DIRECT VERIFY | 194 |
| 84 | ACS | ACCELEROMETER B-CONVERTER 2 | NO DIRECT VERIFY | 054 |
| 85 | HAPS | HAPS HTR/TM MODULE-CONVERTER 1 | NO DIRECT VERIFY | 154 |
| 86 | HAPS | HAPS HTR/TM MODULE-CONVERTER 2 | NO DIRECT VERIFY | 0D4 |
| 87 | HAPS | E/V CMD LOGIC 1-CONVERTER 1 | NO DIRECT VERIFY | 1D4 |
| 88 | HAPS | E/V CMD LOGIC 1-CONVERTER 2 | NO DIRECT VERIFY | 034 |
| 89 | HAPS | E/V CMD LOGIC 2-CONVERTER 1 | NO DIRECT VERIFY | 134 |
| 90 | HAPS | E/V CMD LOGIC 2-CONVERTER 2 | NO DIRECT VERIFY | 0B4 |
| 91 | HAPS | HAPS HEATER GROUP 1 ON | DSC 11 BIT 1=1 | 1B4 |
| 92 | HAPS | HAPS HEATER GROUP 1 OFF | DSC 11 BIT 1=0 | 074 |
| 93 | HAPS | HAPS HEATER GROUP 4 ON | DSC 11 BIT 2=1 | 174 |
| 94 | HAPS | HAPS HEATER GROUP 4 OFF | DSC 11 BIT 2=0 | 0F4 |
| 95 | HAPS | HAPS HEATER GROUP 6 ON | DSC 11 BIT 3=1 | 1F4 |
| 96 | HAPS | HAPS HEATER GROUP 6 OFF | DSC 11 BIT 3=0 | 00C |
| 97 | HAPS | HAPS HEATER GROUP 7 ON | DSC 11 BIT 4=1 | 10C |
| 98 | HAPS | HAPS HEATER GROUP 7 OFF | DSC 11 BIT 4=0 | 08C |
| 99 | HAPS | ARM HAPS HEATER GROUPS 2,3,5 | DSC 11 BIT 5=1 | 18C |
| 100 | HAPS | DISARM HAPS HEATER GROUPS 2,3,5 | DSC 11 BIT 5=0 | 04C |
| 101 | HAPS | HAPS HEATER GROUP 2 ON | DSC 11 BIT 6=1 | 14C |
| 102 | HAPS | HAPS HEATER GROUP 2 OFF | DSC 11 BIT 6=0 | 0CC |
| 103 | PYRO | SOLAR ARRAY DEPLOY-PRIMARY SYS FIRE | STATUS BIT 4=0 | 1CC |
| 104 | PYRO | SOLAR ARRAY DEPLOY-REDUNDT SYS FIRE | STATUS BIT 5=0 | 02C |
| 105 | PYRO | SI DUST COVER DEPLOY-FIRE | EXP. DATA | 12C |
| 106 | PYRO | APOGEE BOOST MOTOR-FIRE | ASC-3 CH.19 | 0AC |
| 107 | HAPS | HAPS HEATER GROUP 3 ON | DSC 11 BIT 7=1 | 1AC |
| 108 | HAPS | HAPS HEATER GROUP 3 OFF | DSC 11 BIT 7=0 | 06C |
| 109 | HAPS | HAPS HEATER GROUP 5 ON | DSC 11 BIT 8=1 | 16C |
| 110 | HAPS | HAPS HEATER GROUP 5 OFF | DSC 11 BIT 8=0 | 0EC |
| 111 | | SPARE | | 1EC |
| 112 | CMD | COMMAND DECODER ON (NOTE 1) | NO.1=SB 20/NO.2=SB 21 =1 | 01C |
| 113 | ACS | FSS NO.1-SENSOR HEAD TOGGLE | DSC 2 BIT 16 (NOTE 4) | 11C |
| 114 | ACS | FSS NO.2-SENSOR HEAD TOGGLE | DSC 3 BIT 16 (NOTE 4) | 09C |

Table 5-2. Impulse Commands (Continued)

| CMD# | SYS | FUNCTION | VERIFICATION | HEX |
|---|---|---|---|---|
| 115 | | SPARE | | 19C |
| 116 | | SPARE | | 05C |
| 117 | | SPARE | | 15C |
| 118 | | SPARE | | 0DC |
| 119 | | SPARE | | 1DC |
| 120 | CMD | COMMAND DECODER ON (NOTE 1) | NO.1=SB 20/NO.2=SB 21 =1 | 03C |
| 121 | | SPARE | | 13C |
| 122 | | SPARE | | 0BC |
| 123 | | SPARE | | 1BC |
| 124 | | SPARE | | 07C |
| 125 | | SPARE | | 17C |
| 126 | | SPARE | | 0FC |
| 127 | | SPARE | | 1FC |

NOTE:

1. IMPULSE COMMANDS 0 THROUGH 47, 64 THROUGH 111, AND ALL SERIAL COMMANDS ARE DISABLED BY IMPULSE COMAND 48. THEY ARE ENABLED BY IMPULSE COMMAND 112. IMPULSE COMMAND 120 ENABLES THE ABOVE COMMANDS ON THE DECODER WHICH IS NOT BEING ADDRESSED.
2. PERMITS BATTERY TO BE ON WITH 3RD ELECTRODE OFF.
3. IF AUTO LOAD REMOVE IS DISABLED, FAULT DETECTORS CAN BE MONITORED ON DSC 13 BIT 1.
4. EACH FINE SUN SENSOR HAS TWO HEADS. BIT=0 SAYS HEAD 1 SELECTED.
   NOTE: THE HEX COLUMN LISTS THE THREE HEX DIGITS THAT CHANGE BETWEEN IMPULSE COMMANDS. SEE COMMAND FORMAT BITS 13-24.



*Reprinted from RCA "Telemetry and Command Manual," (IUE-733-76-101).

## COMMAND SEQUENCES

### CMDSEQ - Start Sequence Build

#### Description

The CMDSEQ directive begins the compilation of a command sequence. Upon entry of this statement, all following command requests are entered into a command sequence table. Relative timing between commands may be specified. When the sequence is completely defined, it is available to be executed by the :SEQ function. CMDSEQ clears any previous sequence that has been built.

#### Format

CMDSEQ

#### Example

See combined example following the :SEQ function.

### ENDSEQ - Stop Sequence Build

#### Description

The ENDSEQ directive defines the end of a command sequence.

#### Format

ENDSEQ

#### Example

See combined example following :SEQ function.

## SEQWAIT - Specify Sequence Wait Time

### Description

The SEQWAIT directive specifies a time delay between commands in a command sequence. The delay is specified in tenths of seconds. This directive may only be used between CMDSEQ and ENDSEQ directives and must immediately precede a directive that inserts a command into the sequence.

### Format

SEQWAIT a

where a = an expression which evaluates to an INTEGER number signifying tenths of seconds (max = 65535).

### Example

See combined example following the :SEQ function.

## :SEQ - Execute Command Sequence

### Description

Initiate the previously defined command sequence. May be used repeatedly as its use does not alter or destroy the sequence. If one or more critical commands exist in the sequence, this will be flagged before initial transmission. Once approved, the entire sequence will be transmitted without further flags, regardless of the total number of critical commands in the sequence.

### Format

:SEQ

### Example

```
100       CALL SEQ1,50          / BUILD COMMAND SEQUENCE
101       :SEQ                  / EXECUTE SEQUENCE
102       WAIT                  / VERIFY SEQ EXECUTION
 :
721 SEQ1  SUBR                  / ENTER CMD SEQ BUILD
```

```
722       CMDSEQ                  / START CMD SEQ BUILD
723       :SIMODE FB,RDHI
724       SEQWAIT 155             / WAIT 15.5 SEC
725       :SIUVC
726       SEQWAIT ARG(1)          / WAIT 5 SEC
727       :SCAN
728       SEQWAIT 5Ø              / WAIT 5 SEC
729       ENDSEQ
730       RETURN                  / RETURN FROM SUBROUTINE
```

This example shows a command sequence being built through a call to a subroutine. Suppose the procedure was executing at line 1ØØ. The procedure will jump, via the CALL statement at line 1ØØ, to the subroutine shown in lines 721-73Ø. There is, as there has to be, a CMDSEQ at the front and an ENDSEQ at the back. Note that at line 726 the SEQWAIT acquires a value passed by the CALL statement. You may use any directive during the sequence build. EXAMINE, WAIT, SNAP, etc., all work normally. After the sequence is built, the procedure returns to line 1Ø1 and executes that sequence.

# SECTION 6. ON-BOARD COMPUTER (OBC) DIRECTIVES

# SECTION 6. ON-BOARD COMPUTER (OBC) DIRECTIVES

The IUE's On-Board Computer (OBC) has three banks of memory. Reference to locations within these banks is done with both octal and decimal numbers. Since the conversion is sometimes confusing, the following tabulation is provided.

| OBC Bank | Octal Locations | Decimal Locations |
|---|---|---|
| Ø | ØØ-Ø7777 | Ø-4Ø95 |
| 1 | Ø1ØØØØ-Ø17777 | 4Ø96-8191 |
| 2 | Ø2ØØØØ-Ø27777 | 8192-12287 |

The ground system consists of four files which pertain to the OBC's contents.

1. Tape Image file,
2. Data Blocks file,
3. Master Image file, and
4. Reconstruct Dump Image file.

The Tape Image and Data Blocks files contain data which can be transmitted to the OBC. If any of this data is transmitted, amd verified, the exact copy of the transmission is copied into the Master Image file. This means that the Master Image file contains what the OBC memories should contain. The Dump Image file contains data transmitted from the OBC, or what the OBC memories do contain. Comparisons can be made between the contents of the Master Image file and the Dump Image file.

## COMMANDS TO OBC

The following set of commands are to the OBC. All OBC commands may be formatted for one of two redundant command paths by issuing the SW OBC,a directive. The first portion of the set are hardcoded. That is, they have no variable field and must be issued exactly as the format shows. Therefore, no example is given as the format is the example.

### :OBC GO - Turn On

Description

Generates a command to the OBC to start the hardware.

### :OBC RESET - Turn Off

Description

Generates a command to the OBC to stop the hardware and to select memory bank No. Ø as the fixed bank.

### :OBC FIX1 - Select Bank 1 as Fixed

Description

Generates a command to the OBC to select memory bank No. 1 as the fixed bank.

### :OBC FIX2 - Select Bank 2 as Fixed

Description

Generates a command to the OBC to select memory bank No. 2 as the fixed bank.

### :OBC DUMP - Dumped Fixed Bank

Description

Generates a command to the OBC to dump the fixed bank of memory. The OBC must be in RESET before this command is given.

## :OBC CMND - General Command

### Description

Generates an OBC command based on the input parameters. On execution, an interrupt 1Ø request is generated.

### Format

:OBC CMND,a,b

where a = an expression which evaluates to a legal integer number, see table 6-1, representing an OBC executive request code.

b = an expression which evaluates to a legal integer number, see table 6-1, representing the data value.

### Example

:OBC CMND,12,7

Table 6-1. Legal :OBC CMND Arguments

| 'a' CODE | 'b' MIN/MAX | 'a' CODE | 'b' MIN/MAX |
|---|---|---|---|
| Ø | Ø/1 | 8 | Ø/2 |
| 1 | Ø/2 | 9 | Ø/1 |
| 2 | 1/32 | 1Ø | Ø/1 |
| 3 | Ø/23 | 11 | Ø/3 |
| 4 | Ø/23 | 12 | Ø/7 |
| 5 | Ø/1 | 13 | Ø/1 |
| 6 | Ø/1 | 14 | Ø/1 |
| 7 | Ø/Ø | 15 | Ø/1 |

## :OBC HLOAD - Load Memory Bank

### Description

Generates commands to prepare the OBC to receive data and then to load the specified OBC bank from the OBC Tape Image file. OBC is reset at beginning of load. Also updates OBC Master Image file with each command successfully transmitted.

### Format

:OBC HLOAD,a<,b>

where a = an expression evaluating to an integer number, Ø, 1, or 2, representing bank number.

b = blank
for load whole bank.
= an expression evaluating to last location of selected bank to load.
= Ø to 4Ø95 for bank Ø.
= 4Ø96 to 8191 for bank 1.
= 8192 to 12287 for bank 2.

### Example

```
:OBC HLOAD,1                    / LOAD OBC BANK 1
```

## :OBC HLD82 - Load Memory Bank (CCIL Unique)

### Description

This directive causes a hardware load of the OBC from the OBC Tape Image File. The difference between HLOAD and HLD82 is in the commands transmitted. HLOAD transmits commands necessary to prepare the OBC to receive the load data before the data commands are transmitted. HLD82 only transmits the data commands.

Format

:OBC HLD82,a<,b>

where a = an expression evaluating an integer number, Ø, 1, or 2, representing memory bank.

b = blank.
= for load whole bank.
= an expression evaluating to last location of selected bank to load.
= Ø to 4Ø95 for bank Ø.
= 4Ø96 to 8191 for bank 1.
= 8192 to 12287 for bank 2.

Example

```
:OBC HLD82,1,7ØØØ                / LOAD PORTION OF BANK 1
```

## :OBC SLOAD - Load Memory Locations

Description

Generates a series of commands to load the OBC memory from contiguous locations on the OBC Tape Image file. Also updates OBC Master Image file after all commands are successfully transmitted.

Format

:OBC SLOAD,a,b<,CKSUM>

where a = an expression evaluating to an integer number $(Ø\text{-}27777)_8$ representing the starting location in both the OBC and the OBC Tape file.

b = an expression evaluating to an integer number $(Ø\text{-}27777)_8$ representing the ending location in both the OBC and the OBC load file. The number must be greater than 'a' but in the same bank.

CKSUM = option to cause update of memory bank checksum.

Example

```
:OBC SLOAD,O'4ØØØ',O'1ØØØØ'      / LOAD 2ND HALF BANK Ø
```

## :OBC PATCH - Load Memory Locations

### Description

Generates a series of commands to load specified data into contiguous OBC memory, starting at a specified location. Also updates OBC Master Image file after all commands are successfully transmitted.

### Format

:OBC PATCH,a,m<,n,...,z>

where a = an expression evaluating to an integer number $(\emptyset\text{-}27777)_8$ representing starting address in OBC memory.

m - z = an expression evaluating to an integer number representing the data to be stored. These may be a maximum of 32 words, each having a maximum of 18 bits. The series of locations cannot cross over memory banks.

### Example

```
:OBC PATCH,O'1Ø5ØØ',O'5Ø3Ø1',O'2Ø71Ø,'O'1Ø1Ø1Ø'
                                / LOAD LOCS 1Ø5ØØ,1Ø5Ø1,1Ø5Ø2
```

## :OBC HDUMP - Dump Selected Bank

### Description

Dumps the selected OBC bank four times through the telemetry system. The DMU must be sampling by ROM2B format, OBC DUMP. This directive will reset the OBC. The dumped data is quality checked and then written out to the OBC Dump file. Since the OBC may loose track of the fixed bank, an :OBC RESET should be sent after the dump. The directive initiates the auto-collection of the data. Conclusion is noted by a message to the event printer.

Format

:OBC HDUMP,a

where a = an expression evaluating to an integer number representing bank number to dump.

= Ø, 1, or 2.

Example

```
:OBC HDUMP,2                    / DUMP OBC BANK 2
```

## :OBC SDUMP - Dump Selected Bank

Description

Dumps the selected OBC bank using the OBC executive program. The DMU must be sampling by ROM2B format, OBC DUMP. Data is written out to the OBC Dump file.

Format

:OBC SDUMP,a

where a = an expression evaluating to an integer number representing bank number to dump.

= Ø, 1, 2.

Example

```
:OBC SDUMP,1                    / DUMP OBC BANK 1
```

## :OBC LDBLK - Load Data Blocks

Description

Loads OBC with specified data block and updates OBC master image file after all commands are successfully transmitted (if they were consecutive address locations).

Format

:OBC LDBLK,a

where a = an expression evaluating to a decimal number (Ø-32) representing data block number.

Example

```
:OBC LDBLK,3                    / LOAD DATA BLOCK 3
```

## MEMORY BANK IMAGE PROCESSING

### OBCLDPRT - Print Tape - Load File

**Description**

Causes an octal print of specified bank of OBC Tape Image file.

**Format**

OBCLDPRT a

where a = an expression evaluating to an integer number representing bank number.

= Ø, 1, or 2.

**Example**

```
OBCLDPRT 2                    / PRINT OBC TAPE FILE BANK 2
```

### OBCMRPRT - Print Load File

**Description**

Causes an octal print of the specified bank of the OBC Master Image file.

**Format**

OBCMRPRT a

where a = an expression that evaluates to an integer number representing bank number.

= Ø, 1, or 2.

## OBCDMPRT - Print Dump File

### Description

Causes an octal print of OBC Reconstructed Dump Image file.

### Format

OBCDMPRT

### Example

```
OBCDMPRT                          / PRINT OBC DUMP FILE BANK 1
```

## OBCLDTYP - Display Selected Load Locations (Not Implemented)

### Description

Ground computer displays selected locations of the OBC tape image file. Display goes to the calling CRT.

### Format

OBCLDTYP a,b

where a = an expression evaluating to an integer number $(\emptyset\text{-}27777)_8$ representing starting location in tape image to display.

b = an expression evaluating to an integer decimal number specifying number of sequential locations to display, maximum of 1Ø.

### Example

```
OBCLDTYP O'2ØØØØ',7               / DISPLAY LOCATIONS Ø2ØØØØ to Ø2ØØØ6
```

## OBCDMTYP - Display Selected Dump Locations (Not Implemented)

### Description

Ground computer displays selected locations of the OBC dump image. Display goes to the calling CRT.

### Format

OBCDMTYP a,b

where a = an expression evaluating to an integer number $(\emptyset\text{-}27777)_8$ representing starting location in dump image to display.

b = an expression evaluating to an integer decimal number specifying number of sequential locations to display, maximum of 1Ø.

### Example

```
OBCDMTYP   0'4ØØ',9              / DISPLAY LOCATIONS Ø4ØØ to ØØ41Ø
```

## OBCLDTAP - Start OBC Tape/Disc Transfer (CCIL Unique)

### Description

This directive reads a magnetic tape mounted on the OL device for the tape header information and then displays the header for the user's inspection. Depending on this information, the load is continued or terminated by the REPLY OBCTAP directive. Tape header is in the following format: TAPE NNNN MMM DD YYY

## REPLY OBCTAP - Process Image Requests (CCIL Unique)

### Description

After the OBC header is displayed, the REPLY OBCTAP directive must be given, with a YES or NO argument. A NO cancels the transfer. A YES causes the transfer of the three memory banks of data on the magnetic tape to the OBC Tape Image file. If the transfer is successful and

the checksum is valid, no further action is necessary. If the checksum is invalid, the message - TAPE CHECKSUM AND COMPUTER CHECKSUM DO NOT COMPARE - is output and an internal flag is set which prohibits loading the file on the spacecraft. You must reply to this message with the REPLY OBCTAP directive, using APPROVE or SKIP as an argument. APPROVE will remove the prohibition of the spacecraft load. SKIP will cause an exit of the tape load procedure leaving the tape image file as unloadable.

Format

REPLY OBCTAP,a

where a = YES
for transferring the mag tape data to load file;
= NO
for no transfer;
= APPROVE
for approving the load file for S/C transmission even through the tape had a bad checksum;
= SKIP
for prohibiting the transmission of the load file to the S/C.

## OBCCKSUM - Calculate Checksum (CCIL Unique)

Description

This directive computes the memory bank checksum for the memory bank locations 'a' to 'b' inclusive. It displays the computer checksum in the following format: CHECKSUM IS NNNNNN.

Format

OBCCKSUM a,b

where a = an expression that evaluates to the starting location to be checked.

b = an expression that evaluates to the ending location to be checked. 'a' and 'b' must be in ascending order and be contained in the same memory bank. Therefore, for bank Ø, 'a' and 'b' range from Ø to 4Ø95 inclusively; for bank 1, 4Ø96 to 8191 inclusively; and, for bank 2, 8192 to 12287 inclusively.

Example

```
OBCCKSUM   4ØØØ,4Ø5Ø          / GET CHECKSUM FOR 51 LOCS
```

## COLLDUMP OBC,ENB - Collect OBC Dump Data (CCIL Unique)

Description

This initiates an automatic collection of the telemetered dump data. When the collection is complete, notification is given via the event printer. This directive assumes that the telemetry format supports the OBC dump data.

Format

```
COLLDUMP OBC,ENB
```

Example

```
COLLDUMP CLEAR                  / CLEAR DUMP AREA
COLLDUMP OBC,ENB                / PROCESS OBC DUMP DATA
```

## SET RECQUAL,OBC - Reconstruct OBC Dump (CCIL Unique)

Description

This directive reconstructs the collected OBC Dump. RECQUAL specifies the quality of minor frames to be used in the reconstruction.

Format

SET RECQUAL,OBC,a

where a = an expression that evaluates to an integer number (1,2,3,4), whose value specifies the reconstruction criteria as listed below:

1. Use all good and questionable minor frames.
2. Use only good and questionable minor frames whose hamming codes are valid.
3. Use only good and questionable minor frames whose sync patterns are correct.
4. Use only good and questionable minor frames whose hamming codes are valid and whose sync patterns are correct.

Example

```
SET RECQUAL,OBC,4                    / USE BEST POSSIBLE DATA
```

## OBCRECON - Construct Best Dump Copy (CCIL Unique)

Description

This directive constructs the best copy of the dumped memory bank from the collected dump data. OBCRECON also assigns a quality indicator to each best copy data word, based on the SET RECQUAL,OBC specification. The quality indicator may be GOOD, BAD, or MISSING. The reconstructed memory bank is saved in the OBC Reconstructed Dump Image File. Only the last reconstructed memory bank image is retained. OBCRECON generates a COLLDUMP CLEAR directive at its conclusion.

Format

OBCRECON a

where a = an expression that evaluates to the memory bank number (Ø, 1, or 2).

Example

```
OBCRECON  2                          / RECONSTRUCT BANK 2
```

## OBCCOMP - Compare Load and Dump Files

### Description

This directive compares the Reconstructed Dump Image file locations with the corresponding Master Image file locations. Non-compared values are printed on the event printer.

### Format

OBCCOMP a,b<,ALL>

where a = an expression that evaluates to the starting location to be compared.

b = an expression that evaluates to the ending location to be compared. 'a' and 'b' must be ascending and must be contained in the same bank. Therefore, for bank Ø, 'a' and 'b' range from Ø to 4Ø95 inclusive; for bank 1, from 4Ø96 to 8191 inclusive; and, for bank 2, from 8192 to 12287 inclusive.

ALL = an option to compare both static and dynamic data words. Default is to compare to only static data words.

### Example

```
OBCCOMP   O'1ØØØØ',O'17777'     / COMPARE BANK 1
```

## OBCCOPY - Transfer Dump Image to Master Image

### Description

The OBCCOPY directive will copy the OBC Reconstructed Dump Image file into one of three parts of the OBC Master Image. OBCCOPY does not update the OBC Master Image file header information for the memory bank.

Format

OBCCOPY <a>

where no argument = copy to memory bank specified in Reconstructed Dump Image file, or

where a = an expression which evaluates to the OBC Master Image memory bank number

= Ø, 1, or 2.

## DATA BLOCK BUILDING

### OBCSEQ - Start OBC Sequence Build

Description

The OBCSEQ directive begins the compilation of an OBC command sequence used in some specific OBC data blocks (14 and 17). After execution of OBCSEQ, all following command requests are entered into an OBC command sequence table, until an OBCEND function is encountered. Relative timing between commands may be specified with the OBCWAIT function. When the OBC sequence is completely defined, it is available for inclusion in an OBC data block. It cannot, however, be executed in a direct manner. OBCSEQ clears any previous sequence that has been built.

Format

OBCSEQ

Example

See combined example following OBCWAIT function.

## OBCEND - Stop OBC Sequence Build

### Description

The OBCEND directive defines the end of the OBC command sequence compilation. See OBCSEQ for further detail.

### Format

OBCEND

### Example

See combined example following the OBCWAIT function.

## OBCWAIT - Specify OBC Sequence Wait Time

### Description

The OBCWAIT directive specifies a time delay between commands in an OBC command sequence compilation. See OBCSEQ for further detail. This directive can only be used between OBCSEQ and OBCEND directives.

### Format

OBCWAIT a

where a = an expression evaluating to an integer number specifying number of tenths of seconds (actually 102.4 msec). Max = 65535.

### Example

For building data block 14:

```
OBCSEQ
:SIMODE   EX,WLC              / EXPOSE WITH WL CAL
OBCWAIT   36000               / EXPOSE 1 HOUR
:SIMODE   STBY                / GO TO STANDBY
OBCEND
```

For building data block 17:

```
OBCSEQ
:IMP 5
:CRU ON,1,16,17,18
OBCWAIT   1Ø                    / WAIT 1 SECOND
:IMP 6
OBCEND
```

## BSEQDB - Build Commanding Data Block (17)

### Description

The BSEQDB directive creates a data block, in a configuration as required for the specified DB, from the OBC command sequence buffer and writes it to the DB file. The BSEQDB directive will abort processing and return an error message if no commands, or more than eight commands, are in the OBCSEQ buffer.

### Format

BSEQDB a

where a = an expression evaluating to an integer number (Ø-32) representing the data block to be generated.

### Example

```
OBCSEQ
:IMP 5
:CRU ON,1,16,17,18
OBCWAIT   1Ø                    / WAIT 1 SECOND
:IMP   6
OBCEND
BSEQDB   17                     / BUILD DB-17
:OBC LDBLK,17                   / LOAD DB-17
```

## OBCDB14 - Build Data Block 14

### Description

The OBCDB14 directive creates data block 14 from the input arguments and the contents of the OBCSEQ buffer, and then writes the block to the DB file. OBCDB14 will examine the OBCSEQ buffer to verify that a proper command sequence exists. That is, it must be a two-command sequence with the serial command address corresponding to the camera mode control command for the selected camera ID.

### Format

OBCDB14 a<,b,c,d>

where a = the ID for one of the four cameras, being either SWR, SWP, LWR, or LWP, and;

b,c,d = one or more of the three DB-14 identifiers FESMTR (FES switch on), MODEXP (time tag alter), and CALLON (cal lamp on). If an identifier is not mentioned, then its corresponding DB bit is set to zero.

### Example

```
OBCSEQ
:SIMODE   EX,WLC              / EXPOSE WITH WL CAL
:OBCWAIT  36000               / EXPOSE 1 HOUR
:SIMODE   STBY                / GO TO STANDBY
OBCEND
OBCDB14   SWR,CALLON          / BUILD DATA BLOCK
:OBC LDBLK,14                 / LOAD DB-14
```

## BPARDB - Build Parameter Set Data Block (12,13,16)

### Description

The BPARDB directive creates a data block by transferring the 18 least significant bits of each word of a specified array into the DB file. Global arrays have been created for DB's 12, FRM(32); 13, PADD(6); and 16, TTACHR(4). Data is entered into the arrays by simple assignment statements.

### Format

BPARDB a,b,c

where a = data block number (Ø-32),
b = name labeling start of data words to be used,
c = number of data words to be used (1-32).

### Example

```
/ BUILD DATA BLOCK 12 TO TELEMETER FRAMES 4 AND 5
INTEGER IDBG(32)
IDBG(1)=4
IDBG(2)=5
DO LOOP K=3,32
LOOP IDBG(K)=-1
BPARDB   12,IDBG,32          / WRITE DB 12 TO FILE
:OBC LDBLK,12                / LOAD DB 12
WAIT                         / PERFORM TESTS
BPARDB 12,FRM,32             / REWRITE NORMAL DB 12 TO FILE
:OBC   LDBLK,12              / LOAD DB 12
```

## OBCDB1Ø - Build Data Block 1Ø

### Description

The OBCDB1Ø directive creates DB 1Ø by transferring the assigned values of the DB variables to the DB file. All assignments are done before OBCDB1Ø is called. Only those fields which are specified by the argument list of OBCDB1Ø are set when the directive is called. All fields

not specifically mentioned, whether or not they have assigned values, are set to zero.

Format

OBCDB1Ø a<,b,c...,z>

where a - z = named global variables associated with DB-1Ø. Refer to table 6-2.

Example

```
/ DO PITCH SLEW AND CAPTURE IN LO GAIN, RAW GYRO
MBO=Ø;MDO=1;MEO=Ø;MJO=1;MK=Ø
SACØ=Ø.5                          / 30 DEGREE SLEW
OBCDB1Ø  MBO,MCO,MDO,MEO,MJO,MK,SACØ
:OBC LDBLK,1Ø                     / LOAD DB 1Ø
```

## OBCLDBLK - Load I&T Data Blocks

Description

The OBCLDBLK directive loads I&T generated data blocks into the data base.

Format

OBCLDBLK <a,...,z>

where no argument = load all data blocks Ø thru 9, 18, and 19, or
where a thru z = an expression which evaluates to a data block number in the ranges of Ø to 9 and 18 to 19.

Example

```
OBCLDBLK 1,5,18                   / LOAD DATA BLOCKS 1, 5, AND 18
```

Table 6-2. OBCDB1Ø Parameters

| | Parameter | Value | Description |
|---|---|---|---|
| Mode 1 - Pitch | MBØ | 1 | Kalman Filter |
| | | Ø | Raw |
| | MCØ | 1 | Gyro & FES Filter |
| | | Ø | Gyro |
| | MDØ | 1 | Slew Control Mode |
| | | Ø | Hold |
| | MEØ | 1 | High Gain |
| | | Ø | Low Gain |
| | MJØ | 1 | Read Slew Command |
| | | Ø | Don't Read |
| | MA | 1 | Attitude With OBC Gyro Trim |
| | | Ø | |
| | MK | Ø | None Trim |
| | | 1 | Ground Control |
| | | 2 | Flight 1st |
| | | 3 | Flight Later |
| | SACØ | >-2/<2 | Slew-Angle Rad Slew |
| | IMXØ | >Ø/<$2^{18}$ | Iterations for Rate |
| | BGØ | >-$2^{-14}$/<$2^{-14}$ | Gyro Drift, rad/sec |
| Mode 2 - Yaw | MB1 | | All same description |
| | MC1 | | as 'Ø' parameters for |
| | MD1 | | Mode 1 |
| | ME1 | | |
| | MJ1 | | |
| | SAC1 | | |
| | IMX1 | | |
| | BG1 | | |

Table 6-2. OBCDB1Ø Parameters (Continued)

| | Parameter | Value | Description |
|---|---|---|---|
| Mode 3 - Roll | MB2 | | All same description |
| | MC2 | | as 'Ø' parameters for |
| | MD2 | | Mode 1 |
| | ME2 | | |
| | MJ2 | | |
| | SAC2 | | |
| | IMX2 | | |
| | BG2 | | |
| Mode 4 - Roll | MF | 1 | Refine Wheel Bias |
| | | Ø | Don't |
| | MG3 | 1 | Add Pitch Wheel Bias |
| | | Ø | Don't |
| | MG2 | 1 | Add Yaw Wheel Bias |
| | | Ø | Don't |
| | MG1 | 1 | Add Roll Wheel Bias |
| | | Ø | Don't |
| | MGØ | 1 | Read Add Wheel Bias |
| | | Ø | Don't |
| | ML | Ø | Disable FES |
| | | 1 | Disable Process |
| | | 2 | Enable |
| | | 3 | Auto |

# SECTION 7. VARIABLE ADDRESS MEMORY (VAM) DIRECTIVES

# SECTION 7. VARIABLE ADDRESS MEMORY (VAM) DIRECTIVES

## COMMANDS

### :VAM - Load VAM (CCIL Unique)

**Description**

Loads a specified VAM Format Group into the Variable Address Memory (VAM) in the DMU. Selection of one of two redundant command paths is accomplished by issuing the SW DMU,a directive. Updates the VAM Master Image file if all commands are successfully transmitted.

**Format**

:VAM a

where a = name of a format which resides in the VAM library.

**Example**

```
SET DECODER,1                  / SWITCH TO COMMAND DECODER 1
XVAM=1                         / ENABLE VAM LOAD
:DMU                           / TRANSMIT
:VAM LWPVAM                    / LOAD NORMAL CAMERA FORMAT INTO VAM
```

## PROCESSING

### COLLDUMP VAM,ENB - Collect Dump Data (CCIL Unique)

**Description**

This directive initiates the collection of telemetry data for a VAM dump.

**Format**

COLLDUMP VAM,ENB

**Example**

```
COLLDUMP CLEAR                 / CLEAR DUMP AREA
COLLDUMP VAM,ENB               / PROCESS VAM DUMP DATA
```

## SET RECQUAL,VAM - Reconstruct Dump (CCIL Unique)

### Description

This directive sets the criteria for reconstruction of the collected VAM dump.

### Format

SET RECQUAL,VAM,a

where a = an expression that evaluates to an integer number specifying the reconstructed criteria:

= 1
for use only good and questionable minor frames;

= 2
for use only good and questionable minor frames whose hamming codes are valid;

= 3
for use only good and questionable minor frames whose sync patterns are correct.

= 4
for use only good and questionable minor frames whose hamming codes are valid and whose sync patterns are correct.

## VAMRECON - Construct Best Image (CCIL Unique)

### Description

The VAMRECON directive constructs the best copy of the VAM from the collected dump data. VAMRECON also assigns a quality indicator to each best copy data word. The quality indicator may be GOOD, BAD or MISSING, as specified by the SET RECQUAL,VAM directive. RECQUAL specifies the quality of minor frames to be used in the reconstruction. The reconstructed VAM is saved in the VAM Reconstructed Dump Image Data Base Group. Only the last reconstructed VAM is retained. VAMRECON generates a COLLDUMP CLEAR directive at its conclusion.

### Format

VAMRECON

## VAMCOMP - Compare Load and Dump Files (CCIL Unique)

### Description

This directive compares the reconstructed VAM with the specified VAM.

### Format

VAMCOMP a

where a = either:

= name of a VAM format (PPR) contained in the VAM Format Data Base Group.

= OBCMASTR

master OBC VAM contained in the VAM Master Image Data Base Group.

= TLMMASTR

master telemetry VAM contained in the VAM Master Image Data Base Group.

## VAMPRT - Print VAM Images (CCIL Unique)

### Description

This directive is to print the VAM Format Data Base Group, the VAM Master Image Data Base Group or the VAM Reconstructed Dump Image Data Base Group.

### Format

VAMPRT a

where a = either:

= the name of a VAM format (PPR) contained in the VAM Format Data Base Group.

= OBCMASTR

the master OBC VAM contained in the VAM Master Image Data Base Group.

= TLMMASTR

the master telemetry VAM contained in the VAM Master Image Data Base Group.

= VAMDUMP
the Reconstructed Dump Image contained in the VAM Reconstructed Dump Image Data Base Group.

## SWITCH - Generate VAM Decom Tables

### Description

This directive generates the VAM decommutation table from the specified VAM. The decommutated tables are stored in the Main Decommutation Table Data Base Group. The telemetry format (ROM or VAM) to be used is specified by the DMU directive.

### Format

SWITCH a

where a = either:

= the name of a VAM format (PPR) contained in the VAM Format Data Base Group.

= TLMMASTR
the master telemetry VAM contained in the VAM Master Image Data Base Group.

## VAMCOPY - Transfer Dump Image to Master Image

### Description

The VAMCOPY directive will copy the VAM Reconstructed Dump Image into one of two parts of the VAM Master Image.

### Format

VAMCOPY <a>

where no argument = copy to the image specified in Reconstructed Dump Image file, or

where a = OBCMASTR
for copy into master OBC VAM Image

= TLMMASTR
for copy into master telemetry VAM Image.

### Example

```
VAMCOPY TLMMASTR                    / COPY VAM DUMP TO TLM VAM
```

# SECTION 8. DATA PROCESSING DIRECTIVES

# SECTION 8. DATA PROCESSING DIRECTIVES

## TELEMETRY

### SET TLMIN - Select Input Stream (CCIL Unique)

#### Description

At system initialization time, both the THS and the DDPS telemetry input device handlers are started. However, neither processor attempts to load any input data into telemetry buffers until the user specifies which stream is to be used. Only one stream may be acquired at any time. This directive selects the appropriate stream.

#### Format

SET TLMIN,a,b

where a = either:

= THS

for addressing the data arriving through the telemetry handling system hardware (direct link); or

= DDPS

for addressing the data arriving through the NASCOM network lines.

b = either:

= ON

for indicating that the specified stream is to be used to load telemetry buffers. The opposite stream will be turned off; or

= OFF

for indicating that the specified stream is not to be used.

#### Example

SET TLMIN,THS,ON

## SET BUFFACT - Select Buffer Size (CCIL Unique)

### Description

The telemetry is grouped into a buffer containing some number of minor frames. The buffer is filled before any data is decommuted and presented to the system. The following directive alters that buffer size.

### Format

SET BUFFACT,a

where a = an expression that evaluates to the power of two which is the number of minor frames to accumulate on one buffer. This expression will be evaluated to between zero and six.

### Example

```
SET BUFFACT,5                    / BUFFER 32 MINOR FRAMES
```

## SET MFFORMAT - Select Decommutation Format (CCIL Unique)

### Description

This directive forces the decommutation of all telemetry minor frames passing the user data quality test to the prescribed format. The last five designators override any indicators of format on the telemetry stream. The AUTO keyword is used to return to automatic format determination algorithm.

### Format

SET MFFORMAT,a

where a = one of the following:

AUTO

FMT1A

FMT1B

FMT2A

FMT2B

VAM

Example

```
SET MFFORMAT,VAM                    / PROCESS VAM FORMAT
```

## SET MFQUAL - Specify Quality Checks (CCIL Unique)

### Description

This directive alters the bits used by telemetry acquisition (see table 8-1) to judge the quality of minor frame data. Two executions of the SET MFQUAL directive are needed to setup the quality check.

### Format

```
SET MFQUAL,a,b
where a = either:
        = GOOD, or
        = DONTCARE.
      b = either:
        = if following GOOD, an expression specifying those bits that
          must be one for the quality to be good, or
        = if following DONTCARE, an expression specifying those bits
          that are to be checked.
```

### Example

```
SET MFQUAL,DONTCARE,2,4,6,8,10,12  / CHECK BITS
SET MFQUAL,GOOD,4,8,12             / CHECK FOR ONES
```

## SET TLMDISP - Select Update Cycle (CCIL Unique)

### Description

This directive sets the number of telemetry cycles between the update of all changed values found on the telemetry stream.

Table 8-1. Telemetry Quality Flags*

| | SCA | FLG8 | FLG7 | FLG5 | FLG4 | FLG3 | FLG2 | FLG1 | SDBO | SDTM | SDE | DEC | QL | CX3M | CX1M | T21 | T20 | EMTM | CBO | TM | NM | ECMCR | BFR | BCE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 4 | 3 | 2 | 1 | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| | | | | | 2 | | | | | | | | | | 1 | | | | | | | | | |

| BIT | |
|---|---|
| 1 | BCE – SET IF BLOCK CODE ERROR DETECTED (HAMING CODE) |
| 2 | BFR – SET IF BIT SYNC OR FRAME SYNC IN REMOTE MODE |
| 3 | ECMCR – SET IF ECMC IN REMOTE MODE |
| 4 | NM – SET IF ECMC IN NORMAL MODE (NOT CONVOLVED) |
| 5 | TM – SET IF ECMC IN THROUGHPUT MODE OR DDPS INPUT IN THROUGHPUT MODE |
| 6 | CBO – SET IF THS BUFFER HAS OVERFLOWED |
| 7 | EMTM – SET IF ECMC OPERATING ON TEST MODE |
| 8-9 | T21, T20 – INDICATES WHICH ECMC TEST IS BEING PERFORMED |
| 10 | CX1M – SET OF ECMC IN CONVOLVED MODE – HARD DECISION |
| 11 | CX3M – SET IF ECMC IN CONVOLVED MODE – SOFT DECISION |
| 12 | QL – SET IF SEQUENTIAL DECODER OUTPUT QUICK LOOK DATA |
| 13 | SEC – SET IF SEQUENTIAL DECODER SUCCESSFULLY OUTPUT DATA |
| 14 | SDE – SET IF SEQUENTIAL DECODER DETECTED AN IRRECOVERABLE ERROR |
| 15 | SDTM – SET IF SEQUENTIAL DECODER IN TEST MODE |
| 16 | SDBO – SET IF SEQUENTIAL DECODER BUFFER OVERFLOWS |
| 17 | FLG1 – SET IF DATA IS BEING INPUT TO THE BIT SYNC |
| 18 | FLG2 – SET IF BIT SYNC IS LOCKED ONTO DATA |
| 19 | FLG3 – SET IF FRAME SYNC DETECTS AN ERROR |
| 20 | FLG4 – SET IF BIT SLIP DETECTED IN FRAME SYNC |
| 21 | FLG5 – SET IF PRECEEDING FRAME WAS NOT TRANSFERRED DUE TO FRAME SYNC LOSS |
| 22 | FLG7 – SET IF TOO MANY FRAME SYNC PATTERN ERRORS WERE DETECTED |
| 23 | FLG8 – SET IF FRAME SYNC IS LOCKED ONTO DATA |
| 24 | SCA – SET IF DATA INVERSION IS BEING PERFORMED |

*Reprinted from Computer Science Corporation 'Control Center Software System Operations Manual" (CSC/SD-76/6055, IM I-76-109).

Format

SET TLMDISP,a

where a = an expression that evaluates the number of telemetry cycles between display updates. Ø≤a≤7.

Example

```
SET TLMDISP,3                    / UPDATE EVERY 3 CYCLES
```

## SET MFSTATIC - Select 'No Update' Limit (CCIL Unique)

Description

This directive resets the static sampling value. The determination of static data is based on a count of the number of minor frames between the acquisition of a sample. If a value for a data point is not received in the proper number of frames, it is classified static. Since the notification of static values is sent to the display system, small values of the static update cycle at high bit rates can be detrimental to system throughput.

Format

SET MFSTATIC,a

where a = an expression which evaluates to an integer denoting the power of two of the number of minor frames between static value determinations, 1≤a≤1Ø; or

= Ø

for no static tag updating.

Example

```
SET MFSTATIC,5                   / CLASSIFY 32 FRAMES AS STATIC
```

Note: Setting MFSTATIC less than BUFFACT has no meaning since the minimum value for MFSTATIC will be BUFFACT minor frames, unless it is a Ø.

## SET THS - Select THS Processing Mode (CCIL Unique)

### Description

This directive controls the parameters sent to the THS at the occurrence of the next error interrupt. The keywords setting up the type of decoding to be performed are mutually exclusive. When NORM is set, it overrides any setting and vice versa. The default value is NORM,RATE,4Ø.
Note that the THS setup parameters are not sent to the THS until an error interrupt is detected. A simulated interrupt can be triggered by the directive SET THS.

### Format

SET THS <,a>

where no argument = a simulated error interrupt, or

where a = either:

= RATE,b

Permits the input of the bit rate to which the THS will be setup when the next error interrupt is signaled. Valid values for the 'b' are 8Ø, 4Ø, 2Ø, 1Ø, 5, 2.5, and 1.25. The directive accepts any valid expression and assigns the bit rate that most closely matches the values mentioned above.

= THRU

Set the throughput mode of THS processing.

= NORM

Reset convolved data operations to handle unconvolved data.

= HCONV

Use hard convolved algorithm for decoding convolved data.

= SCONV

Use soft convolved algorithm for decoding convolved data.

### Example

SET THS,RATE,2.5

## SET DDPS,RATE Select DDPS Rate Parameter

### Description

The SET DDPS,RATE directive selects the telemetry rate for data arriving through the NASCOM lines.

### Format

SET DDPS,RATE,a

where a = an expression evaluating to the selected bit rate.
Valid values for 'a' are 8Ø, 4Ø, 2Ø, 1Ø, 5, 2.5, and 1.25.

### Example

```
SET DDPS,RATE,4Ø                    / SELECT 4ØKB RATE
```

## COLLDUMP - Select Raw Buffer Storage (CCIL Unique)

### Description

This directive is used for collecting telemetry containing OBC dumps, VAM dumps, FES images, and spectrographic images. Data collected by the telemetry processor is maintained in a file as raw telemetry bulk buffers which are stored for later processing.

### Format

COLLDUMP a,b,c,ENB

where a = either:

- = CLEAR
  for clearing dump area;
- = OBC
  indicating that an OBC dump is expected;
- = FES
  indicating that an FES image is expected;
- = VAM
  indicating that a VAM dump is expected;
- = SPEC
  indicating that a spectrographic image is expected.

b = Optional expression evaluating to the minimum number of minor frames to collect. If not specified, this value defaults to 812 for OBC, 1ØØØ for FES, 64Ø for VAM, and 1ØØØ for SPEC.

c = either:

= FMT1A

= FMT1B

= FMT2A

= FMT2B

= VAM

Optional keyword specifying the format on which the telemetry data is collected. No format switching is performed by this directive; however, if the telemetry data is not in the specified format, an error message is printed. If not specified, the data is collected regardless of format.

ENB = Enable the dump. Lack of this parameter leaves dump unenabled. It then must be enabled by a command.

Example

```
COLLDUMP  OBC,812,FMT1A,ENB
```

## CRT OUTPUT

### DISPLAY - Transfer Information to CRT (CCIL Unique)

Description

This directive causes the display of expressions to the CRT screen which executed it. Either the EVENT or DBASEB page should be selected to ensure that the data will remain on the screen long enough to review. It simultaneously causes a print of the data on the event printer.

Format

DISPLAY a<,b,...,z>

where a,b,...,z = a valid CCIL expression that evaluates to an integer, a floating point result, or a quoted text string. The values and text strings are displayed in the appropriate form by type.

Example

```
DISPLAY ARG(1),AF(3)
```

## PAGE - Bring Canned Page to CRT (CCIL Unique)

Description

This directive causes the system to display the requested CRT page on the specified console. See appendix A for page names.

Format

PAGE a<,b>

where a = name of existing page.

b = expression that evaluates to the console number to which the page is being assigned. $1 \leq b \leq 8$

If 'b' is not specified, the page will be displayed at the requesting console.

If 'b' is out of range, the message "NONEXISTENT CONSOLE" is displayed.

Example

```
PAGE DBASEB,3                    / PUT BLANK PAGE ON CONSOLE 3
```

## FREEZE - Inhibit CRT Update (CCIL Unique)

Description

This directive inhibits updates to any data on the CRT screen of the requesting console.

Format

FREEZE

## UNFREEZE - Permit CRT Update (CCIL Unique)

### Description

This directive reinitiates the update of data on the CRT screen of the requesting console.

### Format

UNFREEZE

## PRINTER OUTPUT

### Description

The Sigma 5 computer has two line printers. One printer will be used as an Event Printer (EP) (i.e., events will be documented and monitored to chart the systems progress). The second printer will be used for CRT page shapshots and by programs running in the background.

## DISPLAY - Transfer Information To Printer (CCIL Unique)

### Description

This directive causes the printing of expressions on the event printer. It simultaneously causes the same data to be displayed on the requesting CRT.

### Format

DISPLAY a<,b,...,z>

where a,b,...,z = a valid CCIL expression that evaluates to an integer, a floating point result, or a quoted text string. The values and the text strings are displayed in the appropriate form by type.

### Example

```
DISPLAY   TEMP,RATE(3)/2
```

## SNAP VIRTUAL - Transfer Canned Page To Printer (CCIL Unique)

### Description

This directive causes a print (snapshot) of a named page, that may or may not be currently displayed, to be made on a line printer. See appendix A for page names.

### Format

SNAP VIRTUAL,a

where a = the page name to be snapped on the line printer.

### Example

```
SNAP VIRTUAL,MANTML1            / PRINT MANEUVER TIMELINE
```

## SNAP CONSOLE - Transfer CRT Image To Printer (CCIL Unique)

### Description

This directive causes a print (shapshot) of a currently displayed page to be made on the line printer.

### Format

SNAP CONSOLE<,a>

where no argument = print page from requesting CRT, or

where a = the CRT number whose page is to be printed on the line printer.

### Example

```
SNAP CONSOLE,8                  / PRINT CONSOLE 8 PAGE
```

## SET SPOOL - Select Printer for SNAP's (CCIL Unique)

### Description

This directive alters the printer on which snaps are printed for an individual console. By default, all snap information is output on the high-speed printer (P1).

Format

SET SPOOL,a

where a = P1

for the high-speed printer, and

= P2

for the low-speed printer.

Example

```
SET SPOOL,P1                    / DIRECT SNAPS TO HI-SPEED PRINTER
```

TOP - Top of Page, Low Speed Printer

Description

This directive causes the event printer to skip to top-of-form.

Format

TOP

## STRIPCHART OUTPUT

Two methods, or modes, of stripcharting telemetry data are available - REALTIME and DELAYED. The two modes cannot be mixed.

The REALTIME, or near-real-time, stripcharting facility retrieves up to 1 byte of data for each of 32 stripchart pens at a rate not exceeding 40 times per second. The data is extracted directly from the telemetry minor frames; no conversion or scaling of data is performed. The data to be sent to the stripchart must be specified by minor frame byte position counting from zero to 127. Both maincom and subcom data can be specified. At most, one pen may be assigned to a telemetry data byte as a maincom sample. If a maincom byte is to be assigned to more than one pen it must be described as a subcom assignment that cycles every minor frame (see SUBC paragraph).

*The bulk of the stripchart introductory material is taken verbatim from Computer Sciences Corporation "Control Center Software System Operations Manual' (CSC/SD-76/6055, IM I-76-109).

For status pens, the sampled location is specified by the byte offset of the sample and further modified by a bit number that ranges from Ø through 7 specifying the rightmost and leftmost bits, respectively. The bit number is optional in all cases and defaults to zero (the rightmost bit) if not specified on a status pen assignment.

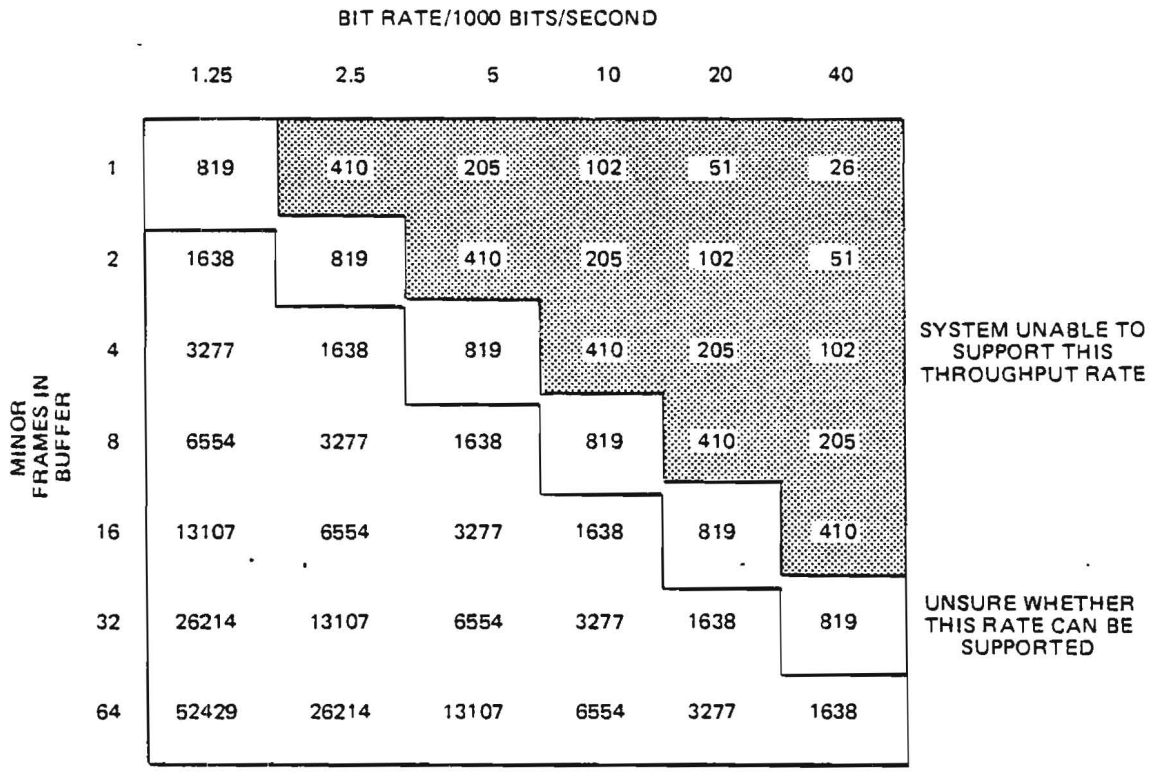
The nature of the stripcharting algorithm causes the data output to be delayed (see table 8-2) by the time necessary for a telemetry buffer to be filled. This time is dependent upon the telemetry bulk buffer size (see SET BUFFACT) and the bit rate (see SET THS/DDPS,RATE). Upon initiation of the stripchart pens, no output is produced until this time period has expired.

The stripcharting algorithms attempt to track buffer factor changes. However, due to the differing buffer sizes of the collected data, the output may not track all changes exactly. No attempt is made to track dynamic bit rate changes. If the bit rate is changed, the stripcharts must be stopped and restarted.

The delayed mode allows retrieval and scaling of decommutated telemetry points. The user specifies the data to be charted by telemetry point name. The options of selecting raw counts or engineering units and of specifying the desired scale is also available.

The output to the pens in this mode is dependent on the TLMDISP parameter. This parameter controls the number of telemetry cycles between delayed-mode stripchart output as well as between display updates. Thus, telemetry buffer size and bit rate changes effect only the rate at which data is output to the pens. The pens need not be stopped and restarted during these changes.

Table 8-2. Stripchart Delay Time (Milleseconds)*




*Reprinted from Computer Sciences Corporation "Control Center Software System Operations Manual' (CSC/SD-76/6055, IM I-76-109).

Stripchart pens are addressed by pen number. These numbers and their locations on the Stripchart Recorders (SCR) are:

| Pen Number | Pen Type | Location |
|---|---|---|
| 1-8 | Analog | Pen 1 is leftmost analog pen on the left recorder |
| 9-16 | Analog | Pen 9 is leftmost analog pen on the right recorder |
| 17-24 | Status | Pen 17 is leftmost status pen on the left recorder |
| 25-32 | Status | Pen 25 is leftmost status pen on the right recorder |

## PEN ON - Activate Pen (CCIL Unique)

### Description

The PEN ON directive starts the output of data, specified in the current pen assignment matrix, to the selected pens.

### Format

PEN ON<,a>

where no argument = activate all pens, or

where a = expression evaluating to a pen number that should be activated.

### Example

```
PEN ON,8                        / START PEN 8
```

## PEN OFF - Deactivate Pen (CCIL Unique)

### Description

The PEN OFF directive stops the output of data to the specified pens.

### Format

PEN OFF<,a>

where no argument = deactivate all pens, or

where a = an expression that evaluates to a pen number that should be deactivated.

### Example

```
PEN OFF                          / STOP ALL PENS
```

## PEN CAL - Set Pen Calibration Value (CCIL Unique)

### Description

The PEN CAL directive allows each pen to be individually calibrated by setting a specified value in the stripchart output list. For calibration, the stripchart mechanism must be on and the particular pen must either be unassigned or turned off.

### Format

PEN CAL,a,b

where a = expression evaluating to the pen number to be calibrated.

b = expression evaluating to the value to be placed on the pen specified. For analog pens, this value may range from Ø through 255. For status pens, the rightmost bit of the resulting value is placed on the status pen.

### Example

```
PEN CAL,17,1
```

## PEN CLEAR - Clear Pen Assignment Matrix (CCIL Unique)

### Description

Ths PEN CLEAR directive will clear the pen assignment matrix, therefore, removing cell data to the pens, setting all output values to zero. The directive may be employed at any time during stripchart setup or operation.

### Format

```
PEN CLEAR                          / CLEAR PEN MATRIX
```

## PEN SAVE - Store Pen Assignments (CCIL Unique)

### Description

The PEN SAVE directive saves the current pen assignment matrix for future use. Up to twelve matrices can be saved.

### Format

PEN SAVE,a<,b>

where a = slot in which matrix is to be stored (1-12).

b = optional text to identify matrix. Text must be contained within single quote marks and may not exceed 40 characters.

### Example

```
PEN SAVE,5,'FES ACQUISITION'
```

## EXAMINE PENMATRX - Display Stored Pen Assignments (CCIL Unique)

### Description

The EXAMINE PENMATRX directive will display the optional text stored with the PEN SAVE directive, along with the slot used.

### Format

```
EXAMINE PENMATRX
```

PEN RESTORE - Copy Stored Pen Assignment to Pen Matrix (CCIL Unique)

Description

The PEN RESTORE directive will transfer a set of stored pen assignments to the current pen assignment matrix.

Format

PEN RESTORE,a

where a = slot number of desired assignment.

Example

```
PEN RESTORE,5                    / ASSIGN FES MATRIX
```

PEN MODE,REALTIME - Set Stripcharting to Realtime Mode (CCIL Unique)

Description

The PEN MODE,REALTIME directive clears the pen assignment matrix, turns off all pens, and sets the mode to realtime. The PEN MAIN and PEN SUBC directives are then the only legal pen assignment directives.

Format

PEN MODE,REALTIME

PEN MAIN - Assign REALTIME Maincom or Supercom (CCIL Unique)

Description

This directive sets up the minor frame byte to pen mapping for maincom and supercom telemetry samples. A maincom telemetry sample is a telemetered item that appears in the same place in the telemetry minor frame for each minor frame received. A supercom telemetry sample is a telemetered item that appears in more than one location per minor frame and is repeated in these same locations for each minor frame.

Format

PEN MAIN,a,m,...,t<,z>

where a = an expression that evaluates to the pen number to be assigned.

m - t = either:

an expression that evaluates to the minor frame byte offset containing the sample or a set of expressions w, x, and y where:

w = an integer that specifies the number of times that the pattern to follow is to be applied.

x = an expression that evaluates to the first minor frame byte offset containing the sample.

y = an expression that evaluates to the number of minor frame bytes between samples.

z = an expression which evaluates to a bit number that modifies the sampling for status pens, if not specified, zero, indicating the rightmost bit (i.e., number of bit in word).

Example

See example following PEN SUBC.

## PEN SUBC - Assign REALTIME Subcom (CCIL Unique)

Description

This directive specifies the pen mapping for subcom data, the location of minor frame, and the byte offset. A subcom telemetry sample is a telemetry item that appears in a fixed location in a minor frame each 'n' minor frames, where 'n' is a power of two.

Format

PEN SUBC,a,b,c,d,<,e>

where a = an expression that evaluates to a pen to be assigned.

b = an expression that evaluates to the minor frame byte offset of the sample.

c = an expression that evaluates to the smallest minor frame counter of a minor frame containing the sample.

d = an expression that evaluates to the number of minor frames between successive samples ('d' must be larger than 'c').

e = an expression that evaluates to the bit to be sampled for status data. Default value is zero, the right-most bit.

Example

Suppose the following items were to be stripcharted.

1. Minor frame counter, byte 6Ø, maincom.
2. Supercom values occurring at minor frame offsets 2, 4, 6, 8, 17, 19, 21, 23.
3. Subcom status point at offset 73 starting on minor frame 5 and occurring every 32 minor frames.

```
PEN MODE,REALTIME            / CLEAR PEN MAPPING, SET MODE
PEN MAIN,1,(6Ø)              / MINOR FRAME COUNTER
PEN MAIN,2,(4(2,2),4(17,2)) / SUPERCOM A CCELB,BUSSV
PEN SUBC,17,73,5,32,5        / SUBCOM STATUS DATA FOR BIT 5
PEN ON                       / START STRIPCHARTS
```

## PEN MODE,DELAYED - Set Stripcharting to Delayed Mode (CCIL Unique)

Description

The PEN MODE,DELAYED directive clears the pen assignment matrix, turns off all pens, and sets the mode to delayed. The PEN directive is then the only legal pen assignment directive.

Format

```
PEN MODE,DELAYED
```

## PEN - Assign Telemetry to SCR in Delayed Mode (CCIL Unique)

### Description

The PEN directive allows stripchart assignment to be made by reference to a named telemetry point (global variable). Three variations of the PEN directive exists: One for charting raw analog counts, one for converted engineering analog valves, and one for raw status counts.

### Format

PEN a,b,RAW<,c,d>

where a = name of a telemetry global variable to be charted on an analog pen.

b = analog pen number on which data is to be placed (1-16).

RAW = selective of raw telemetry counts for display, defaulting to full pen deflection over the count range of Ø to 255.

c = optional modification of RAW's default to Ø counts for minimum scale deflection.

d = optional modification of RAW's default to 255 count for maximum scale deflection.

PEN a,b,ENG,c,d

where a = name of a telemetry global variable to be charted or an analog pen.

b = analog pen number on which data is to be placed (1 to 16).

ENG = selection of engineering units for display.

c = value for minimum scale deflection.

d = value for maximum scale deflection.

PEN a,b

where a = name of a telemetry global variable to be charted on a status pen. If a multi-digit variable is named, the rightmost bit of the raw value (least significant bit) is charted.

b = status pen number on which data is to be placed (19 to 32).

Example

```
PEN MODE,DELAYED                  / CLEAR PEN MATRIX, SET DELAYED MODE
PEN AS1CHØØ,1,RAW                 / CHART AS1CHØØ,RAW,SCALE Ø-255
PEN FORMAT,2,ENG,1,4              / CHART FORMAT, ENGINEERING
                                  / UNITS, SCALE 1-4
PEN MFC,3,RAW,1,64                / CHART MINOR FRAME COUNTER,
                                  / RAW, SCALE 1-64
PEN SAD1,17                       / SOLAR ARRAY DEPLOYMENT
PEN ON                            / START STRIPCHART OUTPUTS
```

# APPENDIX A - PAGE NAMES

| | |
|---|---|
| ACS | Attitude Subsystem |
| ALARM | Alarm Page |
| ASC2 | Test Alarm Page |
| CAMEXPO | Special Camera Page |
| CAM | Camera Status, All Cameras |
| CAM1 | Camera 1 Status |
| CAM2 | Camera 2 Status |
| CAM3 | Camera 3 Status |
| CAM4 | Camera 4 Status |
| COMAN11 | Command Buffer, Page 1 |
| COMAN2 | Command Buffer, Page 2 |
| DATBLK11 | Datablock 11 |
| DBASEB | Blank Page for Work Area |
| DOC | Data Operations and Control |
| EVENT | Event Page |
| FES | FES1 and FES2 Status |
| HWSTATUS | Hardware Status |
| HYDRSTAT | HAPS System |
| MANSEP1 | Unloaded Slews |
| MANSEP6 | Loaded Slews |
| MANTMLNA | Maneuver Time Line |
| MANTMLN1 | Maneuver Time Line |
| MODESH | Logic SH Mode |
| OBC | Onboard Computer |
| OPSYSII | OPSYS Status |
| PAGE | List of Available Pages |
| PRENUT | Precession/Nutation |
| PWRSTAT | Power Status |

| | |
|---|---|
| SISTAT | Scientific Instruments Status |
| SPAVARXA | System Parameter Test Page |
| SYSTAT | Systems/Transfer |
| SYSTATM | Systems/Mission |
| SYSTEMP | All Spacecraft Temperatures |
| TLMCMD | Telemetry and Command Subsystem |
| TLMDBC | OBC Telemetry Buffer |
| TLMPWR | Telemetry Information |