~~2mu~~

S9r

Ground Operations Aerospace Language

GOAL

Final Report

Volume V

Application Studies

```
(NASA-CR-136777)  GROUND OPERATIONS                    N74-15891
AEROSPACE LANGUAGE (GOAL). VOLUME 5:
APPLICATION STUDIES  Final Report
(International Business Machines Corp.)                Unclas
~~76~~ p HC $6.00                  CSCL 09B    G3/08    28969
80
```

31 July 1973

Ground Operations Aerospace Language

GOAL

Final Report

Volume V

Application Studies

Contract NAS 10-6900

Approved by: [illegible]
J. W. Handley, Manager
Systems Programming and
Advanced Programs

IBM Federal Systems Division

31 July 1973

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

1.0 INTRODUCTION

The Ground Operations Aerospace Language (GOAL) was designed to be used by test oriented personnel to write procedures which would be executed in a test environment. A series of discussions between NASA LV-CAP personnel and IBM resulted in some peripheral tasks which would aid in evaluating the applicability of the language in this environment, and provide enhancement for future applications. The results of these tasks are contained within this volume.

The GOAL vocabulary provides a high degree of readability and retainability. To achieve these benefits, however, the procedure writer utilizes words and phrases of considerable length. Brief form study was undertaken to determine a means of relieving this burden. The study resulted in a version of GOAL which enables the writer to develop a dialect suitable to his needs and satisfy the syntax equations. The output of the compiler would continue to provide readability by printing out the standard GOAL language. This task is described in Section 2.0.

Section 3.0 provides the results of a study which identifies and recommends resolution of general problems associated with using engineering units in the GOAL language. A candidate set of units to support the terminology of various engineering disciplines is identified, and methods for input/output scaling of these units to the MKS system are recommended.

Section 4.0 defines validation requirements and techniques for passing parameters to subroutines. Validation procedures are defined for subroutine writers, subroutine users, and for configuration control via the Data Bank. Implementation techniques are defined for the Data Bank and the Compiler.

Several current automated procedures were manually converted to the GOAL language to verify the adaptability of the language to the environment. Section 5.0 discusses this conversion and provides examples of the results.

The final section of the volume, Section 6.0, defines the use of system macros to facilitate frequently required and/or complex GOAL sequences. Several macros were developed and are demonstrated to show feasibility.

SECTION 2.0

GOAL SHORT FORMS

I

2.1 INTRODUCTION

The GOAL compiler has the capability of processing test procedures written in the GOAL language and/or a suitable alternative language. The alternate language may be a 'shorthand' form of GOAL or even a foreign language. The present version of the compiler provides a short form vocabulary for use in writing GOAL test procedures. Table 2-1 contains two lists of all GOAL words and phrases with their corresponding short form alternates. The first list is in the order of the GOAL syntax equations. The second list is in alphabetical order of the GOAL words and phrases.

2.2 GUIDELINES FOR DEVELOPING A PARSABLE ALTERNATIVE VOCABULARY

2.2.1 GOAL Parsing Technique

The parsing of GOAL statements is controlled by the syntax table developed from the MBNF syntax equations. The compiler uses a table guided top-down parsing algorithm. The syntax table guides the parser in scanning statements in the input stream. Permissible constructions are recognized according to the contents of the syntax table. The recognition criterion is simple appearance. If a construction is not recognized, alternate definitions are tested.

EXAMPLE:

```
          <GOAL STMT> = <SYSTEM STMT> |

                        <PROCEDURAL STMT> |

                        <DECLARATION STMT>;
```

The syntax table will guide the parser through all system statements until a permissible construction is identified. If the construction is not recognized as a system statement then the parser will test the construction against the procedural and then the declaration statements.

The technique implies that keyword verbs must be unique in the order of appearance. For instance, if 'AB' were the first system statement keyword verb, then no alternate statement keyword could be composed of 'AB' as the first two characters.

EXAMPLE:

```
          <SYSTEM STMT> = <KEYWORD 1> |

                          <KEYWORD 2> |

                          <KEYWORD 3> ;

          <KEYWORD 1> = 'AB' ;

          <KEYWORD 2> = 'ABC' ;

          <KEYWORD 3> = 'A' ;
```

Keywords 1 and 3 will be recognized by the parser. However, keyword 2 will never be recognized because of the lack of uniqueness in the order of keyword appearance. That is, whenever keyword 2, 'ABC', is parsed it will satisfy the keyword 1 equation.

The following equations show the required order of appearance of keywords for proper parsing.

```
          <SYSTEM STMT> = <KEYWORD 1> |

                          <KEYWORD 2> |

                          <KEYWORD 3> ;

          <KEYWORD 1> = 'ABC' ;

          <KEYWORD 2> = 'AB' ;

          <KEYWORD 3> = 'A' ;
```

The above equations will enable parsing of either of the three keywords because the keywords are now unique in the order of their appearances in the equations.

2.3 IMPLEMENTING AN ALTERNATIVE VOCABULARY

A particular alternative vocabulary is implemented through the GOAL modified Bachus Naur Form syntax equations. More specifically, the implementation is accomplished by substituting or including alternative text in the GOAL vocabulary syntax equations.

There are five groups of syntax equations which contain all of the GOAL vocabulary.

- o System statement Keywords
- o Prefix Keywords
- o Procedural Statement Keywords
- o Declaration Statement Keywords
- o GOAL Words and Phrases

The method of implementing an alternative vocabulary is the same for either group.

The GOAL vocabulary and an alternative vocabulary may appear in the MBNF syntax equations in three forms as follows:

```
     Form 1.  <X> = 'T1'  ;
```

Where X is the symbolic name of a GOAL word or phrase and T1 is the text for the GOAL word or phrase. The Form 1 equation makes no provision for alternatives to the GOAL words and phrases.

```
     Form 2.  <X> = 'T1' | <Z> ;

              <Z> = R1 'T2' R2 ;
```

Where X and T1 are the same as in Form 1, Z is the symbolic name of an alternate for T1. T2 is the alternative text for T1. R1 and R2 are action subroutines, called by the parser, which accomplish the substitution of GOAL words and phrases, T1, for corresponding alternates, T2.

```
     Form 3.  <X> = <Y> | <Z> ;

              <Y> = 'T1' 'S'? ;

              <Z> = R1 'T2' R3 ;
```

Where X, Z, T1, R1, T2 are as previously defined. Y is the symbolic name of an optionally plural GOAL word. R3 is the action routine, called by the parser, which appends the letter S to the GOAL word if the alternative word is pluralized. Care must be taken not to make a word optionally plural whenever the next word in the statement begins with the letter S.

2.4 EXAMPLES OF MBNF EQUATIONS

<u>Form 1</u> is used when there is to be no alternate for the GOAL word or phrase.

```
          <BEGIN> = 'BEGIN' ;
```

<u>Form 2</u> is used when there is an alternate for the GOAL word or phrase.

```
          <BEGIN> = 'BEGIN'| <C001> ;

          <C001> = #5101 'BGN' #5102 ;
```

The Form 2 equation enables parsing of either the word BEGIN or BGN. If the user has specified the compiler directive for substituting GOAL words and phrases for their alternates, then the word BGN will be replaced by BEGIN.

Form 3 is used when there is a short form for an optionally plural GOAL word or phrase.

```
<COLUMNS> = <C077> | <C078> ;

<C077> = 'COLUMN' 'S'?  ;

<C078> = #5101 'C' #5103 ;
```

The Form 3 equation enables parsing of the words COLUMN, COLUMNS, C and CS. If the user has specified the compiler directive for substituting GOAL words and phrases for their alternates, then the words C and CS will be replaced by COLUMN and COLUMNS respectively.

2.5 EXAMPLES OF STATEMENTS USING ALTERNATIVE WORDS AND CONVERSION TO THE GOAL VOCABULARY

a. BGN PGM:

converts to,

BEGIN PROGRAM:

b. STP , S190, S200;

converts to,

STOP AND INDICATE RESTART LABELS STEP190, STEP200;

Note: LABELS is followed by a word beginning with the letter S and therefore cannot be optionally plural.

c.
```
DCL STA TAB (WATER VALVE)
            W 1 R AND 2 CS
            TTL    (OPEN),(CLOSED) WE
            <WVON> , (ON) , (OFF) ;
```

converts to,

```
DECLARE STATE TABLE (WATER VALVE)
            WITH 1 ROW AND 2 COLUMNS
            TITLED  (OPEN),(CLOSED) WITH ENTRIES
            <WVON> , (ON) , (OFF) ;
```

2.6 IMPLEMENTING AN ALTERNATIVE VOCABULARY

To implement an alternative vocabulary one must select a Form 2 or Form 3 equation for each alternate word or phrase to be accommodated by the GOAL compiler. Then by following the procedures in paragraphs 2.3 and 2.4 above the MBNF syntax equations are written to describe a particular vocabulary. These equations, when keypunched, are used to replace corresponding equations in the GOAL Syntax Equation Table. This modified table is then used to generate a uniquely numbered GOAL compiler syntax table. This table will control parsing by the GOAL compiler when the table number is specified to the compiler at compile time, enabling statements written in the alternative vocabulary to be parsed and/or converted to the GOAL vocabulary.

2.7 IMPLEMENTING ERROR MESSAGES

Whenever errors are detected by the GOAL compiler, appropriate messages are listed for the test engineer to assist him in developing syntactically and semantically correct test procedures. These messages are contained in the GOAL error message file and are written in the context of the GOAL vocabulary. For example:

error number 925 states,

KEYWORD NOT FOUND - COLUMNS.

error number 100 states,

INVALID ROW DESIGNATOR OR KEYWORD 'ROW' IS MISSING

When an alternative vocabulary is to be implemented an error message file should be developed in terms of that vocabulary. The GOAL error message file is the baseline for developing error messages for an alternative vocabulary. The error message numbers are not to be changed. It is recommended that a distinct error message file be developed for each alternative to the GOAL vocabulary. The appropriate error file, GOAL or alternative error file, can be selected at compile time for use by the GOAL compiler.

ERROR MESSAGE CARD FORMAT

COLUMN 1, 2 AND 3 - Three digit error number.

COLUMN 5 through 80 - The error message.

2.8 GOAL SHORT FORM EXAMPLE

Figure 2-1 is the GOAL Compiler listing of a test procedure which was written in short form GOAL. This test procedure is a portion of the Instrument Unit Mechanical Systems Procedure for support of IU Stage Power. The test procedure was compiled under the user option to convert from the short form to GOAL.

Table 2-1

SHORT FORM LIST 1

* SYSTEM STATEMENT KEYWORDS

* GOAL	* SHORT FORM
BEGIN	BGN
MACRO	MAC
PROGRAM	PGM
SUBROUTINE	SUB
END	END
EXECUTE	XEC
EXPAND	XPD
EXPAND AND EXECUTE	XAX
FREE	FRE
REPLACE	RPL
USE	USE

* PREFIX KEYWORDS

* GOAL	* SHORT FORM
STEP	STEP
S	S
AFTER	AFR
WHEN	WHN
VERIFY	VFY
IF	IF

Table 2-1
(Continued)

* PROCEDURAL STATEMENT KEYWORDS

* GOAL	* SHORT FORM
ACTIVATE	ACT
APPLY	APA
SEND	SDA
ASSIGN	ASN
AVERAGE	AVG
EVERY	EVY
CONCURRENTLY	CNC
DELAY	DLY
WAIT	WTE
DISABLE	DBL
GO TO	GTO
INHIBIT	INH
ISSUE	ISU
LEAVE	LVE
LET	LET
PERFORM	PER
CRITICAL	CTL
READ	RDE
MEASURE	MSR
DISPLAY	DSP
PRINT	PRT
RECORD	RCD
RELEASE	RLS
REPEAT	REP

Table 2-1
(Continued)

REQUEST	REQ
RESUME	RSM
SET	SET
OPEN	OPN
CLOSE	CLO
TURN ON	TON
TURN OFF	TFF
STOP	STP
TERMINATE	TRM
SYSTEM	SYS
WHEN INTERRUPT	WNT
* DECLARATION STATEMENT KEYWORDS	
DECLARE	DCL
NUMBER	NBR
QUANTITY	QTY
STATE	STA
TEXT	TXT
NUMERIC LIST	RC LST
QUANTITY LIST	QTY LST
STATE LIST	STA LST
TEXT LIST	TXT LST
NUMERIC TABLE	NRC TAB
QUANTITY TABLE	QTY TAB
STATE TABLE	STA TAB
TEXT TABLE	TXT TAB

Table 2-1
(Continued)

* GOAL WORDS AND PHRASES

* GOAL	* SHORT FORM
ALL	ALL
AND	AND
AND INDICATE RESTART LABELS	,
AND SAVE AS	ASA
ARE	ARE
B	B
BETWEEN	BTW
(1) CHARACTERS	CHRS
CLOSED	CLD
(1) COLUMNS	CS
(1) DAYS	DAYS
ELSE	ELSE
ENTRIES	E
ENTRY	E
EQUAL TO	EQ
(1) EXCEPTIONS	XCPS
FALSE	FALSE
FOR	FOR
FROM	FROM
FUNCTIONS	FNS
GREATER THAN	GT
GREATER THAN OR EQUAL TO	GEQ
(1) HRS	HRS

Table 2-1
(Continued)

IS	IS
LESS THAN	LT
LESS THAN OR EQUAL TO	LEQ
(1) MINS	MINS
(1) MSECS	MSECS
NOT	NOT
NOT EQUAL TO	NEQ
OCCURS	,
OFF	OFF
ON	ON
OR	OR
PRESENT VALUE OF	PV
READINGS OF	RO
RETURN	RTN
REVISION	REV
(1) ROWS	RS
(1) SECS	SECS
T	T
TEXT	TXT
THEN	THEN or ,
THROUGH	-
TIMES	TMS
TITLED	TTL
TO	TO

Table 2-1
(Continued)

TRUE	TRUE
UNTIL	TIL
(1) USING MESSAGES	MSGS
WITH	W
WITH A MAXIMUM OF	WMAX
WITH ENTRIES	WE
WITHIN	WIN
X	X

Note (1) - The letter S is optional

Table 2-1
(Continued)

SHORT FORM LIST 2

* ALPHABETICAL LISTING

*	GOAL	* SHORT FORM
	ACTIVATE	ACT
	AFTER	AFR
	ALL	ALL
	AND	AND
	AND INDICATE RESTART LABELS	,
	AND SAVE AS	ASA
	APPLY	APA
	ARE	ARE
	ASSIGN	ASN
	AVERAGE	AVG
	B	B
	BEGIN	BGN
	BETWEEN	BTW
(1)	CHARACTERS	CHRS
	CLOSE	CLO
	CLOSED	CLD
(1)	COLUMNS	CS
	CONCURRENTLY	CNC
	CRITICAL	CTL
(1)	DAYS	DAYS
	DECLARE	DCL
	DELAY	DLY
	DISABLE	DBL
	DISPLAY	DSP

Table 2-1
(Continued)

	ELSE	ELSE
	END	END
	ENTRIES	E
	ENTRY	E
	EQUAL TO	EQ
	EVERY	EVY
(1)	EXCEPTIONS	XCPS
	EXECUTE	XEC
	EXPAND	XPD
	EXPAND AND EXECUTE	XAX
	FALSE	FALSE
	FOR	FOR
	FREE	FRE
	FROM	FROM
	FUNCTIONS	FNS
	GO TO	GTO
	GREATER THAN	GT
	GREATER THAN OR EQUAL TO	GEQ
(1)	HRS	HRS
	IF	IF
	INHIBIT	INH
	IS	IS
	ISSUE	ISU
	LEAVE	LVE
	LESS THAN	LT
	LESS THAN OR EQUAL TO	LEQ

Table 2-1
(Continued)

	LET	LET
	MACRO	MAC
	MEASURE	MSR
(1)	MINS	MINS
(1)	MSECS	MSECS
	NOT	NOT
	NOT EQUAL TO	NEQ
	NUMBER	NBR
	NUMERIC LIST	RC LST
	NUMERIC TABLE	NRC TAB
	OCCURS	,
	OFF	OFF
	ON	ON
	OPEN	OPN
	OR	OR
	PERFORM	PER
	PRESENT VALUE OF	PV
	PRINT	PRT
	PROGRAM	PGM
	QUANTITY	QTY
	QUANTITY LIST	QTY LST
	QUANTITY TABLE	QTY TAB
	READ	RDE
	READINGS OF	RO
	RECORD	RCD
	RELEASE	RLS

Table 2-1
(Continued)

	REPEAT	REP
	REPLACE	RPL
	REQUEST	QT
	RESUME	RSM
	RETURN	RTN
	REVISION	REV
(1)	ROWS	RS
	S	S
(1)	SECS	SECS
	SEND	SDA
	SET	SET
	STATE	STA
	STATE LIST	STA LST
	STATE TABLE	STA TAB
	STEP	STEP
	STOP	STP
	SUBROUTINE	SUB
	SYSTEM	SYS
	T	T
	TERMINATE	TRM
	TEXT	TXT
	TEXT LIST	TXT LST
	TEXT TABLE	TXT TAB
	THEN	THEN or ,
	THROUGH	-

Table 2-1
(Continued)

	TIMES	TMS
	TITLED	TTL
	TO	TO
	TRUE	TRUE
	TURN ON	TON
	TURN OFF	TFF
	UNTIL	TIL
	USE	USE
(1)	USING MESSAGES	MSGS
	VERIFY	VFY
	WAIT	WTE
	WHEN	WHN
	WHEN INTERRUPT	WNT
	WITH	W
	WITH A MAXIMUM OF	WMAX
	WITH ENTRIES	WE
	WITHIN	WIN
	X	X

Note (1) - The letter S is optional

```
 CCCCCCCCCC   OOOOOOOOOOOO  NN        NN  VV        VV  RRRRRRRRRRR   TTTTTTTTTTTT
CCCCCCCCCCCC  OOOOOOOOOOOO  NNN       NN  VV        VV  RRRRRRRRRRRR  TTTTTTTTTTTT
CC        CC  OO        OO  NNNN      NN  VV        VV  RR        RR       TT
CC            OO        OO  NN NN     NN  VV        VV  RR        RR       TT
CC            OO        OO  NN  NN    NN  VV        VV  RR        RR       TT
CC            OO        OO  NN   NN   NN  VV        VV  RRRRRRRRRRRR       TT
CC            OO        OO  NN    NN  NN  VV        VV  RRRRRRRRRRR        TT
CC            OO        OO  NN     NN NN   VV      VV   RR    RR           TT
CC            OO        OO  NN      NNNN    VV    VV    RR     RR          TT
CC        CC  OO        OO  NN       NNN     VV  VV     RR      RR         TT
CCCCCCCCCCCC  OOOOOOOOOOOO  NN        NN      VVVV      RR       RR        TT
 CCCCCCCCCC   OOOOOOOOOOOO  NN         N       VV       RR        RR       TT


                                             AAAAAAAAAA
                                            AAAAAAAAAAAA
                                            AA        AA
                                            AA        AA
                                            AA        AA
                                            AAAAAAAAAAAA
                                            AAAAAAAAAAAA
                                            AA        AA
                                            AA        AA
                                            AA        AA
                                            AA        AA
                                            AA        AA
```

Figure 2-1. (1 of 12)

```
                                                                                    GOAL 01    PAGE    1

                                   GOAL COMPILER SOURCE RECORD LISTING

   RECORD      SOURCE RECORD

      1              BGN PGM (IUCOOL) REV 9;
      2        * CONVERT ;
      3        * TITLE (IU TCP WRITTEN IN A SHORT FORM DIALECT AND CONVERTED TO GOAL);
      4        * DATE (5JUL73) ;
      5              USE (MECMODEL);
      6              USE (TERMINALS);
      7              DCL TXT (PUMPON) WMAX 2 CHRS;
      8
      9              DCL QTY (STAGE INLET PRESS 1), (STAGE INLET PRESS 2) ;
     10
     11              DCL STA TAB (COOLING SYSTEM GN2 FILL ON SWITCH)
     12                                 W 1 R AND 3 CS
     13                                 TTL          (ON),(OFF),(AUTO) WE
     14                                 <MDO 1815>, ON , OFF , OFF ;
     15              DCL STA TAB (WATER CONTROL VALVE CLOSED SWITCH)
     16                                 W 1 R AND 3 CS
     17                                 TTL          (OPEN),(CLOSED),(AUTO) WE
     18                                 <MDO 1814>, OFF  ,   ON   ,  ON ;
     19              DCL STA TAB (IU COOLANT PUMP 1 ON SWITCH)
     20                                 W 1 R AND 3 CS
     21                                 TTL         (ON),(OFF),(AUTO) WE
     22                                 <MDO 1817>, ON , OFF , OFF ;
     23              DCL STA TAB (IU COOLANT PUMP 2 ON SWITCH)
     24                                 W 1 R AND 3 CS
     25                                 TTL          (ON),(OFF),(AUTO) WE
     26                                 <MDO 1358>, ON , OFF , OFF ;
     27              DCL STA TAB (PRESSURE SWITCH ACTIVATED SW)
     28                                 W 1 R AND 3 CS
     29                                 TTL          (ON),(OFF),(AUTO) WE
     30                                 <MDO 1343>, ON , OFF , OFF ;
     31              DCL STA TAB (IU COOLANT PUMP 1 AND 2 OFF SWITCH)
     32                                 W 1 R AND 3 CS
     33                                 TTL          (ON), (OFF), (AUTO) WE
     34                                 <MDO 1342>, ON ,  OFF ,  OFF ;
     35              DCL STA TAB (HIGH PRESSURE REGULATOR ON SW)
     36                                 W 2 RS AND 3 CS TTL
     37                                              (LOW), (HIGH), (AUTO) WE
     38                                 <MDO 490>, OFF ,   ON  ,  OFF ,
     39                                 <MDO 491>, ON  ,   OFF ,  OFF ;
     40
     41              DSP      TXT (.Y.THIS PROCEDURE WILL BRING THE)
     42                       TXT (.Y.GROUND COOLING UNITS AND THE),
     43                       TXT (.Y.IU THERMAL CONDITIONING SYSTEM),
     44                       TXT (.Y.TO OPERATING CONDITION), TO <CRT2-3>;
     45              STP;
     46
     47              PER (GSCU POWER UP) REV 1:
     48              DSP      TXT (.C.BEGIN IU PNEUMATIC PANEL PREPS),
     49                       TXT (.C.USE OIS CHANNEL 154),
     50                       TXT (.C.REPORT TO CUNP WHEN COMPLETE), TO <CRT2-3>;
```

Figure 2-1. (2 of 12)

```
                        GOAL COMPILER SOURCE RECORD LISTING

RECORD     SOURCE RECORD

   51
   52                    $ ON THE IU GROUND PNEUMATICS PANEL ;
   53
   54     S110     RELEASE S 175;
   55          VFY <SYSTEM INLET PRESSURE GAUGE> IS BTW 3300 PSIG AND 6000 PSIG
   56              ELSE DSP XCPS (.R. GN2 SUPPLY NOT 3300 - 6000 PSIG)
   57              TO <CRT2-5> AND STP , S110, S 115 ;
   58     S115     AVG  6 RO  <STAGE INLET PRESSURE GAUGE>
   59                   ASA (STAGE INLET PRESS 1) ;
   60     S120     VFY <MANIFOLD PRESSURE GAUGE> IS LT 100 PSIG
   61                   ELSE DSP XCPS (.R.MANIFOLD PRESSURE NOT L.T. 100 PSI)
   62                   TO <CRT2> AND STP , S120, S 130;
   63     S130     SET (HIGH PRESSURE REGULATOR ON SW) FNS  TO (LOW);
   64              WTE  5  SECS;
   65
   66     S140     VFY  <HIGH PRESSURE REG ON LAMP> IS OFF ELSE DSP XCPS
   67                   (.R.HIGH PRESS REG ON LAMP IS NOT OFF) TO <CRT2>
   68                   AND STP , S 140, S 150;
   69
   70                   $ IF MANIFOLD PRESSURE IS OUT OF TOLERANCE DO S 160;
   71
   72     S150 VFY <MANIFOLD PRESSURE GAUGE> IS BTW 1600 PSIG AND 1800 PSIG
   73                   ELSE DSP XCPS
   74                   (.R.MANIFOLD PRESS IS NOT 1700 +/-100) TO <CRT2-0>
   75                   AND GTO S160;
   76          GTO S 180;
   77     S160     SET (HIGH PRESSURE REGULATOR ON SW) FNS TO (AUTO);
   78              DLY 5 SECS;
   79
   80              SET <REDUNDANT  PRESS REG ON SW> TO ON;
   81              DSP TXT (.G.REDUNDANT PRESS REG ON COMMAND),
   82                  TXT (.G.HAS BEEN ISSUED), TO <CRT2> ;
   83          WTE  5 SECS;
   84
   85          VFY <REDUNDANT PRESS REG ON LAMP> IS ON ELSE DSP XCPS
   86                   (.R.REDUN PRESS REG LAMP IS NOT ON) TO <CRT2>
   87                   AND STP , S 160, S 170 ;
   88     S170 AVG 3 RO <STAGE INLET PRESSURE GAUGE>
   89                        ASA (STAGE INLET PRESS 2);
   90          IF (STAGE INLET PRESS 1) IS GEQ
   91             (STAGE INLET PRESS 2) THEN  GTO  S 180;
   92          DSP TXT (.Y.STAGE INLET PRESSURE INCREASED),
   93              TXT (.Y.HI PRESS REG READING WAS) (STAGE INLET PRESS 1),
   94              TXT (.Y.REDUN PRESS REG READING WAS) (STAGE INLET PRESS 2),
   95              TO <CRT2-0> ;
   96          DSP TXT (.Y.PRESENT VALVE OF STAGE INLET PRESS),
   97                   TO <CRT2-10>;
   98     S175 EVY 3 SECS  CNC PER PGM (STAGE GN2 INLET PRESS) REV 1;
   99          STP , S110, S180;
  100     S180 RLS S175 ;
```

Figure 2-1. (3 of 12)

```
                                                                                  GOAL 01    PAGE   3

                                   GOAL COMPILER SOURCE RECORD LISTING

  RECORD      SOURCE RECORD

    101          SET <SUPPLY SYSTEM ON SW> TO ON;
    102          DLY 5 SECS;
    103
    104          VFY <SUPPLY SYSTEM ON LAMP> IS ON ELSE DSP XCPS
    105                   (.R.SUPPLY SYSTEM ON LAMP IS NOT ON)
    106                   TO <CRT2-3> AND STP, S180, S190;
    107     S190 RLS S195;
    108          VFY <STAGE INLET PRESSURE GAUGE> IS BTW 1600 PSIG AND 1800 PSIG
    109                   ELSE DSP XCPS
    110                   (.R.STAGE INLET PRESS NOT 1600-1800 PSI) TO <CRT2-9>
    111                   AND GTO S194;
    112          GTO S 200;
    113     S194     DSP  TXT (.Y.STAGE INLET PRESSURE),
    114                        TO <CRT2-10>;
    115     S195 EVY 3 SECS CNC PER PGM (STAGE GN2 INLET PRESS) REV 1;
    116          STP , S190, S200;
    117     S200 RLS S195;
    118          DSP TXT (.G.IU PNEUMATICS PANEL PREPS COMPLETE),
    119              TXT (.G.NOTIFY CUNP ON OIS CHANNEL 154), TO <CRT2-0>;
    120          TRM;
    121          END PGM;
```

Figure 2-1. (4 of 12)

```
                        GOAL COMPILER EXPANDED SOURCE STATEMENT LISTING

 STMT      EXPANDED SOURCE STATEMENT

   1          BGN PGM (IUCOOL) REV 9;
   2          USE (MECMODEL);
   3          USE (TERMINALS);
   4          DECLARE TEXT (PUMPON) WITH A MAXIMUM OF 2 CHARACTERS;

   5          DECLARE QUANTITY (STAGE INLET PRESS 1), (STAGE INLET PRESS 2) ;

   6          DECLARE STATE TABLE (COOLING SYSTEM GN2 FILL ON SWITCH)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED     (ON),(OFF),(AUTO) WITH ENTRIES
                                 <MDO 1815>, ON , OFF , OFF ;
   7          DECLARE STATE TABLE (WATER CONTROL VALVE CLOSED SWITCH)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED     (OPEN),(CLOSED),(AUTO) WITH ENTRIES
                                 <MDO 1814>, OFF  ,   ON   ,  ON ;
   8          DECLARE STATE TABLE (IU COOLANT PUMP 1 ON SWITCH)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED    (ON),(OFF),(AUTO) WITH ENTRIES
                                 <MDO 1817>, ON , OFF , OFF ;
   9          DECLARE STATE TABLE (IU COOLANT PUMP 2 ON SWITCH)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED     (ON),(OFF),(AUTO) WITH ENTRIES
                                 <MDO 1358>, ON , OFF , OFF ;
  10          DECLARE STATE TABLE (PRESSURE SWITCH ACTIVATED SW)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED     (ON),(OFF),(AUTO) WITH ENTRIES
                                 <MDO 1343>, ON , OFF , OFF ;
  11          DECLARE STATE TABLE (IU COOLANT PUMP 1 AND 2 OFF SWITCH)
                                 WITH 1 ROW AND 3 COLUMNS
                                 TITLED     (ON), (OFF), (AUTO) WITH ENTRIES
                                 <MDO 1342>, ON ,  OFF ,  OFF ;
  12          DECLARE STATE TABLE (HIGH PRESSURE REGULATOR ON SW)
                                 WITH 2 ROWS AND 3 COLUMNS TITLED
                                           (LOW), (HIGH), (AUTO) WITH ENTRIES
                                 <MDO 490>, OFF ,   ON  ,  OFF ,
                                 <MDO 491>, ON  ,   OFF ,  OFF ;
```

Figure 2-1. (5 of 12)

GOAL COMPILER EXPANDED SOURCE STATEMENT LISTING

```
STMT      EXPANDED SOURCE STATEMENT

                                   $ ********** BEGIN OPERATING STEPS ********** ;

 13          DISPLAY  TEXT (.Y.THIS PROCEDURE WILL BRING THE)
                     TEXT (.Y.GROUND COOLING UNITS AND THE),
                     TEXT (.Y.IU THERMAL CONDITIONING SYSTEM),
                     TEXT (.Y.TO OPERATING CONDITION), TO <CRT2-3>;
 14          STOP;

 15          PERFORM (GSCU POWER UP) REVISION 1;
 16          DISPLAY  TEXT (.C.BEGIN IU PNEUMATIC PANEL PREPS),
                     TEXT (.C.USE OIS CHANNEL 154),
                     TEXT (.C.REPORT TO CUNP WHEN COMPLETE), TO <CRT2-3>;

                                $ ON THE IU GROUND PNEUMATICS PANEL ;

 17     STEP110     RELEASE STEP 175;
 18          VERIFY <SYSTEM INLET PRESSURE GAUGE> IS BETWEEN 3300 PSIG AND 6000 PSIG
                 ELSE DISPLAY EXCEPTIONS (.R. GN2 SUPPLY NOT 3300 - 6000 PSIG)
                TO <CRT2-5> AND STOP AND INDICATE RESTART LABELS STEP110, STEP 115 ;
 19     STEP115     AVERAGE 6 READINGS OF <STAGE INLET PRESSURE GAUGE>
                      AND SAVE AS (STAGE INLET PRESS 1) ;
 20     STEP120     VERIFY <MANIFOLD PRESSURE GAUGE> IS LESS THAN 100 PSIG
                      ELSE DISPLAY EXCEPTIONS (.R.MANIFOLD PRESSURE NOT L.T. 100 PSI)
                     TO <CRT2> AND STOP AND INDICATE RESTART LABELS STEP120, STEP 130;
 21     STEP130     SET (HIGH PRESSURE REGULATOR ON SW) FUNCTIONS TO (LOW);
 22               WAIT  5  SECS;

 23     STEP140     VERIFY <HIGH PRESSURE REG ON LAMP> IS OFF ELSE DISPLAY EXCEPTIONS
                     (.R.HIGH PRESS REG ON LAMP IS NOT OFF) TO <CRT2>
                     AND STOP AND INDICATE RESTART LABELS STEP 140, STEP 150;

                              $ IF MANIFOLD PRESSURE IS OUT OF TOLERANCE DO S 160;

 24     STEP150 VERIFY <MANIFOLD PRESSURE GAUGE> IS BETWEEN 1600 PSIG AND 1800 PSIG
                      ELSE DISPLAY EXCEPTIONS
                      (.R.MANIFOLD PRESS IS NOT 1700 +/-100) TO <CRT2-0>
                      AND GO TO STEP160;
 25          GO TO STEP 180;
 26     STEP160     SET (HIGH PRESSURE REGULATOR ON SW) FUNCTIONS TO (AUTO);
 27               DELAY 5 SECS;

 28               SET <REDUNDANT  PRESS REG ON SW> TO ON;
 29               DISPLAY TEXT (.G.REDUNDANT PRESS REG ON COMMAND),
                     TEXT (.G.HAS BEEN ISSUED), TO <CRT2> ;
 30          WAIT  5 SECS;

 31          VERIFY <REDUNDANT PRESS REG ON LAMP> IS ON ELSE DISPLAY EXCEPTIONS
                     (.R.REDUN PRESS REG LAMP IS NOT ON) TO <CRT2>
                     AND STOP AND INDICATE RESTART LABELS STEP 160, STEP 170 ;
 32     STEP170 AVERAGE 3 READINGS OF <STAGE INLET PRESSURE GAUGE>
```

Figure 2-1. (6 of 12)

```
                         GOAL COMPILER EXPANDED SOURCE STATEMENT LISTING

STMT      EXPANDED SOURCE STATEMENT

                           AND SAVE AS (STAGE INLET PRESS 2);
 33           IF (STAGE INLET PRESS 1) IS GREATER THAN OR EQUAL TO
                 (STAGE INLET PRESS 2) THEN  GO TO STEP 180;
 34           DISPLAY TEXT (.Y.STAGE INLET PRESSURE INCREASED),
                 TEXT (.Y.HI PRESS REG READING WAS) (STAGE INLET PRESS 1),
                 TEXT (.Y.REDUN PRESS REG READING WAS) (STAGE INLET PRESS 2),
                  TO <CRT2-0> ;
 35           DISPLAY TEXT (.Y.PRESENT VALVE OF STAGE INLET PRESS),
                        TO <CRT2-10>;
 36       STEP175 EVERY 3 SECS  CONCURRENTLY PERFORM PROGRAM (STAGE GN2 INLET PRESS) REVISION 1;
 37           STOP AND INDICATE RESTART LABELS STEP110, STEP180;
 38       STEP180 RELEASE STEP175 ;
 39           SET <SUPPLY SYSTEM ON SW> TO ON;
 40           DELAY 5 SECS;

 41           VERIFY <SUPPLY SYSTEM ON LAMP> IS ON ELSE DISPLAY EXCEPTIONS
                   (.R.SUPPLY SYSTEM ON LAMP IS NOT ON)
                   TO <CRT2-3> AND STOP AND INDICATE RESTART LABELS STEP180, STEP190;
 42       STEP190 RELEASE STEP195;
 43           VERIFY <STAGE INLET PRESSURE GAUGE> IS BETWEEN 1600 PSIG AND 1800 PSIG
                        ELSE DISPLAY EXCEPTIONS
                        (.R.STAGE INLET PRESS NOT 1600-1800 PSI) TO <CRT2-9>
                        AND GO TO STEP194;
 44           GO TO STEP 200;
 45       STEP194     DISPLAY TEXT (.Y.STAGE INLET PRESSURE),
                           TO <CRT2-10>;
 46       STEP195 EVERY 3 SECS CONCURRENTLY PERFORM PROGRAM (STAGE GN2 INLET PRESS) REVISION 1;
 47           STOP AND INDICATE RESTART LABELS STEP190, STEP200;
 48       STEP200 RELEASE STEP195;
 49           DISPLAY TEXT (.G.IU PNEUMATICS PANEL PREPS COMPLETE),
                 TEXT (.G.NOTIFY CUNP ON OIS CHANNEL 154), TO <CRT2-0>;
 50           TERMINATE;
 51           END PROGRAM;
```

Figure 2-1. (7 of 12)

```
                                       INTERNAL NAME CROSS-REFERENCE LISTING

          INTERNAL NAME                   TYPE      SIZE     DEFINED AT      REFERENCED AT

(AUTO)                                  COLUMN     #ROWS       0006        0026
(CLOSED)                                COLUMN     #ROWS       0007        ** UNREFERENCED **
(COOLINGSYSTEMGN2FILLONSWITCH)          STATE      01X03       0006        ** UNREFERENCED **
(HIGHPRESSUREREGULATORONSW)             STATE      02X03       0012        0021 0026
(HIGH)                                  COLUMN     #ROWS       0012        ** UNREFERENCED **
(IUCOOLANTPUMP1AND2OFFSWITCH)           STATE      01X03       0011        ** UNREFERENCED **
(IUCOOLANTPUMP1ONSWITCH)                STATE      01X03       0008        ** UNREFERENCED **
(IUCOOLANTPUMP2ONSWITCH)                STATE      01X03       0009        ** UNREFERENCED **
(LOW)                                   COLUMN     #ROWS       0012        0021
(OFF)                                   COLUMN     #ROWS       0006        ** UNREFERENCED **
(ON)                                    COLUMN     #ROWS       0006        ** UNREFERENCED **
(OPEN)                                  COLUMN     #ROWS       0007        ** UNREFERENCED **
(PRESSURESWITCHACTIVATEDSW)             STATE      01X03       0010        ** UNREFERENCED **
(PUMPON)                                TEXT       00001       0004        ** UNREFERENCED **
(STAGEINLETPRESS1)                      QUANTITY   00001       0005        0019 0033 0034
(STAGEINLETPRESS2)                      QUANTITY   00001       0005        0032 0033 0034
(WATERCONTROLVALVECLOSEDSWITCH)         STATE      01X03       0007        ** UNREFERENCED **
```

Figure 2-1. (8 of 12)

```
                    STATEMENT LABEL CROSS-REFERENCE LISTING

 LABEL     DEFINED AT       REFERENCED AT

S 0110       0017        0018 0037
S 0115       0019        0018
S 0120       0020        0020
S 0130       0021        0020
S 0140       0023        0023
S 0150       0024        0023
S 0160       0026        0024 0031
S 0170       0032        0031
S 0175       0036        0017 0038
S 0180       0038        0025 0033 0037 0041
S 0190       0042        0041 0047
S 0194       0045        0043
S 0195       0046        0042 0048
S 0200       0048        0044 0047
```

Figure 2-1. (9 of 12)

```
                          FUNCTION DESIGNATOR CROSS-REFERENCE LISTING

        DATA BANK NAME              DATA BANK NUMBER

(MECMODEL)                              0001
(TERMINALS)                             0002
```

Figure 2-1. (10 of 12)

```
                                    FUNCTION DESIGNATOR CROSS-REFERENCE LISTING
        FUNCTION DESIGNATOR                    TYPE           ADDRS  DATA BNK     REFERENCED AT
<CRT2>                                  SYSTEM I/O            00002    0002    0020 0023 0029 0031
<CRT2-0>                                SYSTEM I/O            00200    0002    0024 0034 0049
<CRT2-10>                               SYSTEM I/O            00210    0002    0035 0045
<CRT2-3>                                SYSTEM I/O            00203    0002    0013 0016 0041
<CRT2-5>                                SYSTEM I/O            00205    0002    0018
<CRT2-9>                                SYSTEM I/O            00209    0002    0043
<HIGHPRESSUREREGONLAMP>                 SENSOR DISCRETE       00292    0001    0023
<MANIFOLDPRESSUREGAUGE>                 SENSOR ANALOG         00123    0001    0020 0024
<MD01342>                               LOAD DISCRETE         00904    0001    0011
<MD01343>                               LOAD DISCRETE         00892    0001    0010
<MD01358>                               LOAD DISCRETE         00902    0001    0009
<MD01814>                               LOAD DISCRETE         00860    0001    0007
<MD01815>                               LOAD DISCRETE         00916    0001    0006
<MD01817>                               LOAD DISCRETE         00900    0001    0008
<MD0490>                                LOAD DISCRETE         00320    0001    0012
<MD0491>                                LOAD DISCRETE         00248    0001    0012
<REDUNDANTPRESSREGONLAMP>               SENSOR DISCRETE       00087    0001    0031
<REDUNDANTPRESSREGONSW>                 LOAD DISCRETE         00602    0001    0028
<STAGEINLETPRESSUREGAUGE>               SENSOR ANALOG         00148    0001    0019 0032 0043
<SUPPLYSYSTEMONLAMP>                    SENSOR DISCRETE       00028    0001    0041
<SUPPLYSYSTEMONSW>                      LOAD DISCRETE         00572    0001    0039
<SYSTEMINLETPRESSUREGAUGE>              SENSOR ANALOG         00111    0001    0018
```

Figure 2-1. (11 of 12)

```
                              GOAL COMPILER DIAGNOSTIC SUMMARY

WARNINGS.

    THE FOLLOWING INTERNAL NAMES WERE UNREFERENCED :

      (CLOSED)                           (COOLINGSYSTEMGN2FILLONSWITCH)    (HIGH)
      (IUCOOLANTPUMP1AND2OFFSWITCH)      (IUCOOLANTPUMP1ONSWITCH)          (IUCOOLANTPUMP2ONSWITCH)
      (OFF)                              (ON)                              (OPEN)
      (PRESSURESWITCHACTIVATEDSW)        (PUMPON)                          (WATERCONTROLVALVECLOSEDSWITCH)

END OF DIAGNOSTICS.

TOTAL NUMBER OF SOURCE RECORDS:  121
TOTAL NUMBER OF STATEMENTS:   51
TOTAL NUMBER OF WARNINGS:   12
TOTAL NUMBER OF ERRORS :    0
HIGHEST CONDITION CODE WAS  4
```

Figure 2-1. (12 of 12)

SECTION 3.0

ENGINEERING UNITS TECHNIQUES

3.1 PURPOSE OF DOCUMENT

This document represents an effort to define the future role of engineering units in the GOAL language. The advantages and disadvantages will be discussed and the necessary ground rules and conventions will be identified.

3.1.1 Role of Engineering Units

The present use of engineering units is for readability only. No attempt has been made to automatically scale units or check for proper dimensional operations. As to the future of engineering units, this document will identify a recommended system of engineering units and evaluate methods for incorporating the required processing techniques into the GOAL Compiler. Items that affect an Operating System will also be identified.

3.1.2 Advantages

There are several advantages to be gained by incorporating engineering units into the GOAL System. The major advantages are associated with readability, automatic unit scaling, and usage validation. Other benefits include validation of subroutine parameters, and the validation of console inputs to the on-line system.

3.1.3 Disadvantages

In order for engineering units to be properly scaled and validated, additional restrictions must be imposed on the procedure writer. To ensure clear and unambiguous meaning of GOAL statements using units, comprehensive and rigorous tests must be made by the compiler. Once a procedure writer has decided to use engineering units, he must continue using them to the completion of the procedure.

3.1.4 Ground Rules

This document will identify the conventions and guidelines required to implement engineering units in the GOAL Language. Problem areas related to this effort will be discussed.

3.2 TECHNICAL APPROACH

The analysis of engineering units techniques in the GOAL Language will be presented according to the following outline:

3.2.1 System of Units

A candidate set of engineering units is provided to support operations involving electrical, mechanical, and thermal terminology. This set is identified in Table 3-1. It is described in further detail in Section 3.3.

3.2.2 Compiler and Operating System Ground Rules

The use of engineering units is discussed for the various GOAL Language Statements in which they occur. Assumptions regarding real time operations are described. This analysis is given in Section 3.7.

3.2.3 Potential Problem Areas

Problem Areas related to the use of engineering units in the GOAL Language are described in Section 3.12.

3.2.4 Implementation Techniques

A phased approach for implementation of engineering units with the GOAL Compiler is given in Section 3.13.

3.3 SYSTEM OF UNITS

3.3.1 Conventional Systems

The conventional systems, English and Metric, are discussed and a technique is described for conversion to an internal working unit system.

3.3.2 English Systems and Metric Systems Unit Definitions

In any logical system of units it is necessary at the beginning to assume certain units arbitrarily. In each of the commonly used systems there are three fundamental units. The fundamental quantities in the international scientific system are length, mass, and time. In the English system the fundamentalquantities are length, force, and time. A comparison of two forms of the scientific system and one of the English system is as follows:

Quantity	MKS System	CGS System	ENGLISH (f lbm s) System
Length	Meter	Centimeter	Foot
Mass	Kilogram	Gram	Lb Mass
Density	Kilogram/Meter^3	Gram/Centimeter^3	Lb Mass/Foot^3
Force	Newton	Dyne	Poundal
Work (energy)	Joule	Erg	Foot Poundal
Power	Watt	Erg/Sec	Poundal/Sec
Time	Second	Second	Second

As shown in the table the differences between the three systems, other than the assumption of the arbitrary units, is the choice of the fundamental quantities. In CGS and MKS the unit of mass is a fundamental unit while that of force is a derived one.

3.3.3 Conversion Techniques

A variable expressed in the units of one system can be converted to equivalent units in another system. The conversion is accomplished by using a constant relationship between the two systems as a multiplier or divisor. Using the MKS system as a baseline, the conversion constants for the CGS and f lbm s systems are as follows:

Quantity	MKS	CGS	f lbm s
Length	1 Meter	10^2 CM	3.281 ft
Mass	1 Kilogram	10^3 Gram	2.205 lbm
Density	1 K/M^3	10^{-3} GM/cm^3	62.43x10^{-3} lbm/ft^3
Force	1 Newton	10^5 dyne	7.015 poundal
Work (Energy)	1 Joule	10^7 erg	23.02 ft poundal
Power	1 Watt	10^7 erg/sec	23.02 ft poundal/sec

A simple conversion example using the quantity length is as follows:

Convert 6 feet to meters

6 feet = 6 feet ÷ 3.281 feet/meter = 1.829 meters

The conversion technique can be easily applied to convert units from various systems to units of a single system.

3.4 SPECIALIZED SYSTEMS

The principal system units discussed thus far have been those of the Mechanical System. This section will include those units associated with the Electrical and Thermal Systems.

3.4.1 Electrical Systems

There are eight or ten different systems of electrical and magnetic units which are in common use. Each system is based on a choice of a constant of proportionality in an experimentally verified physical law. For the purpose of this paper we will limit our discussion to three of the more common systems. The CGS electrostatic system (esu) units are derived from Coulomb's law which says that the force, f, between two electrical charges, q_1 and q_2, separated by a distance r in empty space is directly proportional to the product of the charges and inversely proportional to the square of the distance between them.

$$f = K\, q_1\, q_2/r^2$$

where K is a constant chosen as unity and dimensionless.

The CGS electromagnetic system (emu) units are derived from the law of attraction between currents. If two currents of magnitude I_1 and I_2 flow in long parallel wires, separated by a distance d, they attract each other with a force per unit length given by

$$f = K\, I_1\, I_2/d$$

where K is a constant chosen as 2 and dimensionless.

The two systems discussed thus far have used CGS mechanical units. By using MKS mechanical units, we are able to define a system of units that coincide closely with the practical system of units which grew up in the nineteenth century. The volt, ampere, henry, farad, and ohm are all units of the MKS system.

Using the MKS system of electrical units as a baseline, the conversion constants and the units associated with each are illustrated in the following table:

Quantity	MKS	esu electrostatic	emu electromagnetic
Charge	1 Coulomb	2.998×10^9 Statcoulomb	.1 Abcoulomb
Potential Difference	1 Volt	3.336×10^{-2} Statvolt	10^8 Abvolt
Current	1 Ampere	2.998×10^9 Statampere	.1 Abampere
Resistance	1 Ohm	1.1126×10^{-12} Statohm	10^9 Abohm
Power	1 Watt	10^7 erg/sec	10^7 erg/sec
Inductance	1 Henry	1.1126×10^{-12} Stathenry	10^9 cm
Capacitance	1 Farad	8.988×10^{11} cm	10^{-9} Abfarad

3.4.2 Mechanical Systems

The mechanical units have been discussed previously. (See the table of MKS, CGS, and English units in Section 3.3.3.

3.4.3 Thermal Systems

Thermal measurements involve the specification of temperature. Two temperature scales are in common use and both are defined in terms of measurements made with a mercury thermometer and both having the freezing and boiling point of pure water at standard atmospheric pressure as the fixed points. The Centigrade scale uses 0^{o} and 100^{o} while the Fahrenheit scale uses 32^{o} and 212^{o}. The scales are extended by the Kelvin or Absolute temperature scale and the Rankine temperature scale. The Kelvin scale is independent of the properties of any particular substance except that the difference between the boiling and freezing point of water will be 100^{o}K. The Zero value is the lowest limit temperature that can be approached but never reached. The Rankine scale corresponds to the Kelvin scale, but is based on absolute zero of the Fahrenheit system. Using the MKS ^{o}C System as the baseline, the relationships among thermal units are as follows:

Quantity	MKS ^{o}C	MKS ^{o}K	CGS^{o}K	f lbms ^{o}F	f lbms ^{o}R
(Temperature Difference)	1^{o}C	1^{o}K	1^{o}K	1.80^{o}F	1.80^{o}R
Temperature	X^{o}C	(273.16+X)^{o}K	(273.16+X)^{o}	F(32+9X/5)^{o}F	(491.7+9X/5)^{o}F
Energy	1 joule	1 joule	10^{7}erg	9.478X10^{-4} BTU	9.478X10^{-4} BTU

3.4.4 Compatibility Between Systems

Compatibility between units of the same systems (Mechanical, Electrical, and Thermal), can be maintained if the units are scaled to a common internal working unit base. Working units will be discussed in Section 3.6.

3.5 AMBIGUITIES

Ambiguities can be created by the use of engineering units if the units are used only part of the time, if they are used incorrectly, or if the user doesn't know what internal working units are being used.

Ambiguities can also be created by the order in which arithmetic operations are performed. This problem will be discussed in greater detail in Section 3.12 (Potential Problem Areas)

3.5.1 Mass vs. Force Conventions

This ambiguity arises in the specification of Mass or Force in the English system of units. If the pound unit is used a compiler will not be able to determine if it is pounds force or pounds mass.

Specific engineering unit symbols will have to be defined for mass and force units in order to provide the GOAL procedure writer with meaningful results.

3.5.2 On, Closed, True vs. Off, Open, False

These operators have been discussed at great length and it is generally agreed that the ambiguity is created by the Mechanical/Electrical System definitions. As with force and mass a convention must be established regarding their use.

3.5.3 Absolute vs. Relative Temperature and Pressure

The Thermal System units involve, in addition to the mechanical fundamental quantities of mass, length, and time (m, l, and t), the specification of temperature (Θ). The present list of engineering units in the Syntax diagram handbook contains the capability for relative or absolute pressure, but does not have the capability for temperature expression. Absolute measurements are with respect to the null condition, while relative measurements are expressed as the difference between two limits, (i.e., 100^{o}A absolute is very cold, 100 A^{o} relative is the difference between the freezing and boiling point of water). Additional units will have to be defined for the representation of Absolute and Relative temperature if they are to be included in the list of legal Engineering units.

3.6 WORKING UNITS

If the GOAL System is to provide the Procedure Writers with the capability to use different engineering units within the same statement, all units will have to be converted to a common denominator. This suggests the use of a system of internal working units. In selecting a set of internal working units it is highly desirable that the system consists of units that are common to Mechanical, Electrical, or Thermal Systems, and are of sufficient size to handle both large and small units. The system that seems best suited to meet this criteria is the MKS system. This system is larger than the CGS system and is about the same size as the English System. There is a distinct advantage in the electrical system units in the MKS system. Nearly all of these quantities evolved during the nineteenth century and units such as the volt, ampere, henry, farad, and ohm are already in the MKS system. The MKS is already the standard system in most all but the English speaking countries today. Some of these countries are converting to the metric system and others are studying the problem. This seems to be the logical system of internal working units for the GOAL System. The procedure writers should be free to use any system of units that they desire, but if automatic scaling of units and usage validation are to be included in the GOAL System, it is recommended that the MKS system units be adopted as the internal working unit.

3.7 COMPILER AND OPERATING SYSTEM GROUND RULES

This section identifies the various ways that engineering units can be used in the GOAL statements.

3.7.1 Quantity Declaratives

These statements are used to identify specific internal names as quantity variables for the GOAL Compiler. The quantity declaration will enable the compiler to validate all statements that use the variable.

3.7.1.1 Initialized Quantity Variables

When a quantity variable is declared it may be assigned initial values. These are expressed as a numeric value followed by some engineering units. These units are used to determine the conversion required to obtain the initial values in the MKS internal working system.

3.7.1.2 Uninitialized Quantity Variables

If a quantity variable is not assigned an initial value it may still be given engineering units which will be used to validate its usage. In this case the numeric value is omitted in the DECLARE statement. This will enable the compiler to validate the statements that use the variable.

3.7.1.3 Dimensionless Quantities

If a quantity variable is not assigned engineering units, it is considered to be a dimensionless quantity and it is assigned 'null' units.

These special quantities will be discussed in Section 3.12.

3.8 ARITHMETICAL EXPRESSIONS

3.8.1 Balanced Dimensions

Arithmetic operations must be performed using engineering units with compatible dimensions. It must be legal to perform arithmetic operations with all types of units; however, the resulting units must be equal to the declared quantity units for the variable to the left of the expression. For expressions that perform either addition or subtraction it is a simple matter to verify that the dimensions are the same. For example, it is legal to add or subtract units whose fundamental quantities of length, mass, or time (l, m, t) are identical. The problem occurs when multiplication and division are performed within an expression. These two arithmetical operations create composite units.

3.8.2 Composite Units

When unlike units are multiplied or divided the resulting composite units may or may not be legal units. The creation of composite units will depend upon the order in which arithmetical expressions are evaluated. This problem will be discussed in Section 3.12.

3.9 COMPARISON TESTS

3.9.1 Compatible Comparands

All comparison tests should be performed on like comparands. This is a continuation of the current GOAL compiler rule that unlike Function Designators cannot be used in comparison tests. Quantity variables used in comparison tests will have to have the same fundamental quantities, but not necessarily the same units. For example, it should be legal to compare inches to feet because the same fundamental quantity, length (1) is common to both variables.

3.9.2 Required Scaling

If internal working units are in a common system such as MKS, then the GOAL Compiler and online executive system can perform the required scaling for quantity variables. If variables, other than quantities, are used for comparison tests or I/O operations, the procedure writer will be responsible for proper scaling.

3.10 ANALOG I/O

3.10.1 Calibration and Scaling

If the engineering unit theme is to be followed throughout the GOAL system, analog values should be automatically scaled for I/O operations. In order to convert the analog value to the proper value, the executive system must have access to the calibration data for measurements and the engineering unit identifiers for each measurement. With this data, the programs could use actual units rather than 0-5 volts or some other scale.

3.11 KEYBOARD AND DISPLAY I/O

3.11.1 Validation and Scaling

Keyboard input can be validated by checking engineering units specified for input parameters. If the GOAL procedure is expecting Volts for input the units can be tested and scaled before the parameter is passed to the program. Output can be handled in the same way except that the units could be optional. If no units were specified, the executive could scale to the MKS unit and output the data. If units were specified, the units will be converted to that specified.

3.12 POTENTIAL PROBLEM AREAS

3.12.1 Special Quantities

Special consideration will have to be made for dimensionless quantities. Items such as specific gravity, specific heat, ¶, e and others which may require special unit designations when automatic scaling and usage validation are to be performed. The potential problem area is associated.

with mixing numeric data or constants with declared quantities. The compiler will be unable to verify arithmetic expressions if there are no restrictions placed upon the variables allowed in the expressions.

3.12.2 Composite Units

Composite units may be generated during the evaluation of arithmetic expressions or when arithmetic calculations are performed over the course of several GOAL statements. The composite units which are generated during the course of evaluation of an arithmetic expression may be the most difficult to implement. These operations will be dependent upon the order of variables as coded by the procedure writer. The entire expression may have to be evaluated and the resulting units compared to the defined units for the variable to the left of the expression.

A method will have to be identified which will allow a GOAL Procedure writer to identify units which are not included in the accepted list of engineering units. A suggested method for defining composite units is shown in Table 3-3.

3.12.3 Temporary Storage of Data

Temporary storage of data should not cause any problems with engineering units since all variables must be declared. The GOAL Procedure writer will have to be aware of the problems identified with composite units when temporary storage locations are defined.

3.12.4 Resolution and Precision

By examination of conversion factor tables for conversion from other systems to the MKS system, six fractional decimal places should be sufficient for units which are expressed in "micro" units. The degree of accurary will probably be determined by the computer word size on which the GOAL system is implemented. Six fractional decimal places should be the smallest number considered. The number of integer digits required must be capable of supporting the "mega" unit. This requires at least 6 digits plus the significant digits. This would imply a minimum of 9 or 10 digits to the left of the decimal. These requirements are within the capabilities of the floating point registers of many of the larger computer systems.

3.13 IMPLEMENTATION TECHNIQUES

3.13.1 Basic Methods

The GOAL Compiler as it exists at the present time is considered the basic system. The engineering units listed in the GOAL Syntax Diagram Handbook may be used to enhance the readability of a GOAL Procedure. A cursory validation is performed by the Compiler to insure that the engineering units used are defined. If an engineering unit is not defined in the GOAL Compiler, it will be flagged as an error. The basic method consists of a definition of all legal engineering units, but values are not scaled to compatible units and operations are not validated.

3.13.2 Automatic Scaling

Automatic scaling of units to internal working units is the next suggested step for the implementation of engineering units. The internal working unit system recommended in Section 3.6 is the MKS System. In addition to recognizing the various types of engineering units as is presently done in the basic system the quantities will be scaled up or down in order to correspond to MKS Units. This operation will be performed during compilation, but the on-line system would have to recognize that all units were MKS. No validating of operations would be performed and it would be possible to add volts to amps and get PSI for a result.

3.13.3 Dimensional Validation

The third phase of implementation is a continuation of Phase Two. Automatic scaling would be performed as suggested in Section 3.13.2. In addition, unit usage will be validated by the compiler.

A usage validation technique can be implemented using three fundamental quantities previously identified, length, mass, and time (l, m, t). The three fundamental quantities would be sufficient if either the mechanical or electrical systems were to be implemented separately; however, some of the fundamental quantities are identical in the two systems. If we consider the permeability of a vacuum, μ, as a fourth fundamental quantity in the electrical system, we can then distinguish between the mechanical and electrical systems. There is one other dimension in the thermal system which is variable and that is temperature, Θ.

If the five defined fundamental quantities are used to define each engineering unit in the GOAL system, all arithmetic operations can be validated by addition, subtraction, or comparison of the subscripts of the fundamental quantities. The operations would be performed according to the following rules:

1. Addition/Subtraction - the values of the subscripts of the five fundamental quantities must be <u>identical</u> for the operation to be legal.

2. Multiplication - the values of the five fundamental quantities subscripts of the multiplier will be <u>added</u> to those of the multiplicand to form the subscripts of the product. The resulting five fundamental quantity values of the product must be identical to those of one of the engineering units defined in the GOAL system.

3. Division - the values of the five fundamental quantity subscripts of the divisor will be <u>subtracted</u> from those of the dividend to form the subscripts of the quotient. The resulting fundamental quantity values of the quotient must be identical to those of one of the engineering units defined in the GOAL System.

With these rules, the user must be aware of the sequence of multiple operations. Intermediate units may be required for operations that are performed over several statements.

The validation can be performed by reducing the operations to a Polish notation string and performing the indicated operations on the fundamental quantity vectors during the compilation phase. Inconsistencies can be detected at this time. Table 3-2 contains the GOAL Engineering units expressed as the five fundamental quantities. Table 3-3 contains examples of unit validation using this technique.

ENGINEERING UNITS

FUNCTION TYPE	BASIC UNIT	$X10^3$	$X10^6$	$X10^{-3}$	$X10^{-6}$
VOLT	VOLT	KILOVOLT	MEGAVOLT	MILLIVOLT	MICROVOLT
CURRENT	AMPERE			MILLIAMP	MICROAMP
FREQUENCY	HERTZ	KILOHERTZ	MEGAHERTZ		
RESISTANCE	OHM	KILOHM	MEGAOHM		
INDUCTANCE	henry			Millihenry	Microhenry
CAPACITANCE	farad				Microfarad
POWER	WATT	KILOWATT		Milliwatt	Microwatt
PRESSURE	lbs/in^2				
	millibars				
	in of Hg				
	millimeters of Hg				
DISTANCE	inch				
	foot				
	meter	kilometer	megameter	millimeter	micrometer
	nautical mile				

TABLE 3-1

TABLE 3-1 (Continued)

ENGINEERING UNITS

FUNCTION TYPE	BASIC UNIT	$X10^3$	$X10^6$	$X10^{-3}$	$X10^{-6}$
VELOCITY	feet/sec				
	meters/sec				
	knot				
	mach. no.				
ANGLE	degree				
	arc min				
	arc sec				
	radian				
	revolution				
TEMPERATURE	degrees centigrade				
	degrees fahrenheit				
	degrees kelvin				
MASS	kilogram			gram	
	slug				

ENGINEERING UNITS

FUNCTION TYPE	BASIC UNIT	X10^{3}	X10^{6}	X10^{-3}	X10^{-6}
DENSITY	K/M^{3}			gm/cm^{3}	
	Slug/ft^{3}				
FORCE	Newton				
	poundal				
WORK	Joule				

TABLE 3-1 (Continued)

TABLE 3-2

DIMENSIONAL FORMULAE

The dimensional formulae are represented by the three fundamental quantities, length, mass, and time (l, m, t) plus the two additional defined quantities temperature (θ) and the dielectric constant of a vacuum (μ).

Unit Name	Equivalent Units
LENGTH	l
MASS	m
TIME	t
TEMPERATURE	θ
AREA	l^2
VOLUME	l^3
VELOCITY	lt^{-1}
ACCELERATION	lt^{-2}
DENSITY	ml^{-3}
FLOW	l^3t^{-1}
FORCE	mlt^{-2}
PRESSURE	$ml^{-1}t^{-2}$
WORK and ENERGY	ml^2t^{-2}
POWER (WATTS)	ml^2t^{-3}
VISCOSITY	$ml^{-1}t^{-1}$
KINEMATIC VISCOSITY	l^2t^{-1}
SURFACE TENSION	mt^{-2}
ROTARY POWER	l^{-1}

TABLE 3-2 (Continued)

Unit Name	Equivalent Unit
QUANTITY OR CHARGE	$\mu^{-1/2}m^{1/2}l^{1/2}$
CURRENT	$\mu^{-1/2}m^{1/2}l^{1/2}t^{-1}$
POTENTIAL	$\mu^{1/2}m^{1/2}l^{3/2}t^{-2}$
RESISTANCE	$\mu l t^{-1}$
VOLUME RESISTIVITY	$\mu l^{2}t^{-1}$
MASS RESISTIVITY	$\mu m l^{-1}t^{-1}$
VOLUME CONDUCTIVITY	$\mu^{-1}l^{-2}t$
MASS CONDUCTIVITY	$\mu^{-1}m^{-1}l t$
CAPACITANCE	$\mu^{-1}l^{-1}t^{2}$
INDUCTANCE	μl
THERMOELECTRIC POWER	$\mu^{1/2}m^{1/2}l^{3/2}t^{-2}\theta^{-1}$
FLUX OF MAGNETIC INDUCTION	$\mu^{1/2}m^{1/2}l^{3/2}t^{-1}$
MAGNETIC FIELD INTENSITY	$\mu^{-1/2}m^{1/2}l^{-1/2}t^{-1}$
MAGNETIC POTENTIAL	$\mu^{-1/2}m^{1/2}l^{1/2}t^{-1}$
RELUCTANCE	$\mu^{-1}l^{-1}$
MAGNETIC INDUCTION	$\mu^{1/2}m^{1/2}l^{-1/2}t^{-1}$

TABLE 3-3

EXAMPLE FORMULAE

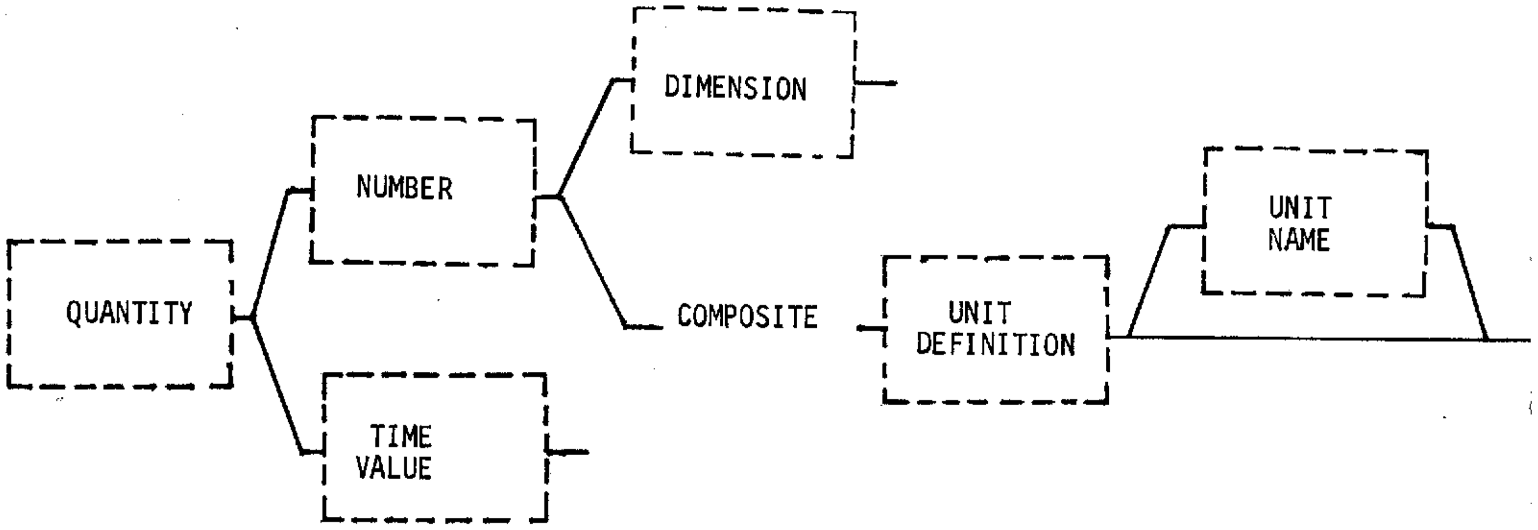
PRESSURE	=	FORCE ÷ AREA
$ml^{-1}t^{-2}$	=	$mlt^{-2} - l^{2}$
$ml^{-1}t^{-2}$	=	$ml^{-1}t^{-2}$
POWER	=	CURRENT x POTENTIAL
$ml^{2}t^{-3}$	=	$\mu^{1/2}m^{1/2}l^{3/2}t^{-2} + \mu^{-1/2}m^{1/2}l^{1/2}t^{-1}$
$ml^{2}t^{-3}$	=	$ml^{2}t^{-3}$
VELOCITY	=	DISTANCE ÷ TIME
lt^{-1}	=	$l - t$
lt^{-1}	=	lt^{-1}
AREA	=	LENGTH x WIDTH
l^{2}	=	$l + l$
l^{2}	=	l^{2}

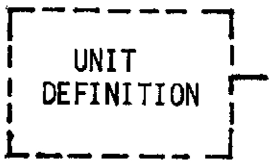
COMPOSITE UNIT EXAMPLE

DENSITY (ml^{-3})	=	MASS (m) ÷ VOLUME (l^{3})
COMPOSITE (ml^{-2})	=	MASS (m) ÷ AREA (l^{2})
DENSITY (ml^{-2})	=	COMPOSITE(ml^{-2}) ÷ HEIGHT (l)

TABLE 3-3 (Continued)

COMPOSITE UNIT DEFINITION SYNTAX DIAGRAM





A combination of the fundamental quantities (m, l, t, θ and μ) that describe the engineering unit being defined.

SECTION 4.0

SUBROUTINE PARAMETER VALIDATION

4.1 PURPOSE OF DOCUMENT

This report will identify the role of the subroutine in the GOAL language. Off-line validation of system subroutine calling sequence will be discussed, and real time error conditions will be identified.

4.2 TECHNICAL APPROACH

Validation criteria will be established which will identify the requirements for various operations associated with the subroutine. Validation procedures will be defined for the subroutine writer, the subroutine user, and for the group responsible for the system data bank. Implementation techniques will be discussed for the GOAL Compiler System Data Bank and On-line Operations.

4.3 VALIDATION CRITERIA

4.3.1 Compilation Time

During the compile of a GOAL subroutine call, the subroutine revision level should be verified, the number and type of parameters certified and in some instances parameter values should be tested for limits.

4.3.2 Execution Time

Some error conditions can be anticipated, but nothing can be done by the off-line system to prevent them from occurring. This type of error is possible because some parameters will be calculated during execution of a GOAL program and the only way to find the problem is to execute the program.

Error recovery from real time error conditions can be accomplished several different ways. Present day operating systems handle the real time error in one of two ways. The program and all steps associated with the program are terminated or a completion code is passed to the program and the program makes its own decision as to what should be done. An extension of the latter method is an error return in the calling sequence for the subroutine where there may or may not also be a completion code. In a real time checkout environment, the decision to continue or terminate should probably be left to the program.

4.4 VALIDATION PROCEDURES

4.4.1 Role of the Data Bank

A special data bank, hereafter referred to as the System Data Bank, should be available to provide the linkage for subroutine parameter validation. It is felt that the subroutine object module should not reside in the System Data Bank, but that a description of the number, type, and size of parameters be included with the subroutine name and revision level.

4.4.2 Role of the Compiler

The GOAL Compiler will be able to verify Subroutine calling sequences by using the descriptive information contained in the System Data Bank. The System Data Bank would be used by the compiler without requiring the USE/FREE data bank statements in the GOAL program.

An additional benefit to be derived from this type of system is that the compiler could create/update a file which contained a list of every program that used each system subroutine. This information would be useful in determining the impact of a subroutine error or a subroutine change. The restriction could also be made that in order for a system subroutine to be successfully compiled, it must be described in the System Data Bank.

4.5 IMPLEMENTATION TECHNIQUES

4.5.1 Data Bank Update

A separate configuration control group should be responsible for maintaining the System Data Bank. Inputs to this group should be Subroutine name, number, type, and limits for each parameter.

4.5.2 GOAL Compiler

The Compiler can be used to validate all system subroutine data once the subroutine descriptive data has been loaded into the System Data Bank. Some limited capability should probably be provided for subroutines that are not system subroutines or subroutines that are not defined in the System Data Bank.

4.5.3 Compilation Output Data

This area could have a special routine that performs parameter validation before executing a subroutine. The only type of validation that should be required at this time is parameter limit validation. It would probably be more reasonable to perform this type of validation in the system subroutine.

SECTION 5.0

SELECTED APPLICATIONS

I

5.0 INTRODUCTION

A study was conducted to analyze the overall applicability of the GOAL language to projected launch vehicle/space vehicle ground checkout requirements. This analysis was accomplished by manually converting selected current ATOLL and machine language programs to GOAL, and extrapolating the results from this language conversion. This conversion and analysis was performed as viewed from the test engineer standpoint.

5.1 OBJECTIVES

The primary objective of this study was to verify that the GOAL language is capable of performing the tasks required for the efficient ground checkout of launch vehicle/space vehicle systems. Emphasis was placed on ascertaining the applicability and utility of this language in converting test engineers' requirements directly into automated, self-documenting procedures. The major points of detailed analysis centered upon:

- Applicability - can the required objectives defined in the test engineers' specifications be accomplished utilizing the GOAL language.
- Utility - is the high level procedural language GOAL the logical or preferred choice for performing those tasks associated with automated launch vehicle checkout.
- Adaptability - is the GOAL language adaptable to the various disciplines, environments, and test facilities encountered in checkout testing.
- Reliability, flexibility, and efficiency - does GOAL contain the inherent characteristics to provide reliability in coding and execution, the flexibility to handle widely varying checkout requirements, while retaining efficiency in procedure generation and documentation.

5.2 APPROACH

GOAL is a high level, engineer oriented language designed to be used by test engineers to write automated test and checkout procedures directly from test specifications and requirements. In order to verify the capability of this language to fulfill previously defined objectives, three existing checkout programs were converted to the GOAL language. Two of these programs, IAED and IATS, were written in the ATOLL language, while the third GE01 was written in machine language for the RCA-110A computer. These programs were selected for several reasons. First, IATS and IAED are both ATOLL programs executed during the final launch sequence. It can be assumed that there will be considerable commonality between these countdown sequences and those proposed for future systems. Second, both of these ATOLL programs are integrated programs, that is, they cover all stages as well as spacecraft. This assures that test specifications for the widest possible variety of applications are being tested by these programs.

Thirdly, both of these ATOLL programs have been updated constantly, and use many ATOLL operators, such as ARITH that were not available during prior ATOLL releases. Fourth, these programs are both quite interactive with the operator, giving ample opportunity to verify this technique in GOAL. Finally, these programs freely use subroutines, allowing study of parameter passing and other techniques.

The machine language program, GE01 was included in order to study the applicability of GOAL to checkout programs that were too specialized in nature to be written in the ATOLL language. Many techniques, formerly available only to the machine language programmer, could be tested using the GOAL instruction repertoire.

The initial approach to the conversion of the ATOLL programs was the one to one approach. That is, for every ATOLL statement there would be one or a corresponding group of GOAL statements. This allowed verification whether all ATOLL statements could be converted directly into GOAL. Once confidence with the overall GOAL language structure was gained, variations from the one to one technique were introduced. GOAL statements were used replacing many ATOLL instructions to test the efficiency of the GOAL language. Various table techniques were also experimented with in order to gain a fuller understanding of the language. No attempt was made to 100% verify the conversions from ATOLL to GOAL, as size limitations precluded an overall compile on the existing GOAL system and with execution of the program not possible even if a compile could be obtained.

The conversion of the machine program GE01 to GOAL was straight forward. Descriptive symbolic names were introduced for the program variables, with the names being chosen so as to be inherently obvious as to the variables replaced.

Some operations in both ATOLL and RCA110A machine language could not be performed directly in GOAL. Others required assumptions as to the executive or implementation system that could not be justified at this time. For those areas where coding the statement in GOAL was not appropriate for these reasons, comment cards were included to indicate those instructions not converted. The statement numbers used in all GOAL statements are identical to those in the source programs, with the exception of those ATOLL steps utilizing six significant digits, which could not be handled by the current GOAL Compiler.

5.3 TECHNICAL SUMMARY

The conversion of major portions of the IAED, IATS, and GE01 programs to the GOAL language establishes that GOAL, in its present form, is capable of handling most tasks required for launch vehicle ground checkout. The GOAL language proved relatively easy to learn and use, and the english language nature of GOAL reduces program documentation to a simple task. In general, GOAL, when combined with certain assumptions as to its interface with the test system, should be able to handle directly any task that may be required.

5.3.1 Language Syntax and Structure

The checkout language must adapt to the task at hand, that of ground checkout of launch vehicles in an engineering environment. It is considered desirable that checkout programs be written to conform to the engineering discipline when possible, rather than to utilize highly specialized programming techniques. Typical ATOLL programs provide test checkout by commanding the system under test to an initial configuration, issuing a specific stimulus to that system, and comparing test results with predetermined conditions. Proper verification of this data usually results in program progression, while an error or abnormal condition usually results in the display of an error message and a program branch to an error routine.

These tasks can be accomplished quite effectively using GOAL, as is shown in the following example.

```
$** ST 124 INVERTER POWER ON;

STEP02 SET (ST124) FUNCTIONS TO (TEST);

       SET <MD01313> , <MD01314> TO ON FOR 1 SEC;

STEP03 VERIFY <MDI2121> IS ON WITHIN 100 MSEC ELSE DISPLAY EXCEPTIONS
       (*IU ST124 INVERTER DID NOT COME ON WHEN COMMANDED)
       TO <CRT-A6> , <CRT-A7> , <CRT-A9> AND GO TO STEP2013;

STEP04 DISPLAY TEXT (IU ST124 INVERTER ON NORMALLY)
       TO <CRT-A6> , <CRT-A7> , <CRT-A9>;
```

This sequence is typical of current ATOLL application programs. The attributes of GOAL are quite obvious when reading this example. The test progression is apparent in this example, even to one with no training in GOAL. No logic ambiguity exists, for example, as to what messages will be displayed when testing on the state of MDI2121.

Many instructions proved extremely useful and versatile during this study. The VERIFY prefix was the most widely used, as when combined with an execution statement allowed the testing of an external sensor, the display of an appropriate message, and the performance of the required response, all without program branching. This type of combined operation greatly simplifies program logic and materially reduces chance for error. The second most widely used operator was 'SET', and, when used for multiple discretes, or with its time option proved versatile and uncomplicated. In general, the operator set was applicable to solving the problem at hand with little need to comment. In certain respects, however, the study revealed some areas where further study might lead to language enhancements.

5.3.1.1 External References. A symbolic name that references hardware items or data not locally located in a particular GOAL program are denoted by the use of angle brackets, such as <MDI1313>, while names that are local to that program are denoted by parenthesis (ABCD). This is quite useful, as external references are immediately obvious, and one level of redundancy is provided by the required verification of the usage of an external reference by its assignment in the data bank. The GOAL language uses separate operators for internal and external data. The VERIFY prefix provides an example of this. When internal names are referenced, the IF/THEN option must be utilized, requiring positive logic and deleting the output exception capability. When testing external designators, the full capabilities of this powerful prefix are brought to bear, including the output exception compounded with a response action, and either positive or negative logic. It would seem useful if the full power of this VERIFY prefix were made available to all types of variable testing. These comments apply equally well to most other operations where a distinction is made between internal names and external references. Further advantages could be obtained by reductions in the syntax set, and hence improve the ease of learning/using the language.

5.3.1.2 Syntax Documentation. The GOAL language is thoroughly documented by the use of syntax diagrams. These diagrams represent a new technique and quite adequately document the language. However, in a language with this large and complex instruction set, the number of syntax diagrams used tends to increase the learning time for the language. It might be possible to reduce the number of required syntax diagrams by:

a. Eliminating obvious diagrams, such as LETTER and NUMBER

b. Combining others, such as LIST NAME, COLUMN NAME, and PROGRAM NAME

c. Reorganizing the syntax handbook into a users guide, introducing a few ground rules, such as "are leading zeros allowed on step numbers?" that may not be obvious or easily represented by syntax notation.

5.3.1.3 Subroutines and Macros. These are programming aids more familiar to the programmer than the engineer. Subroutines however, have been successfully used by engineers in many ATOLL procedures. One feature of subroutines that is not natural to the engineer is the method of parameter passing. The concept most easily understood by an engineer is the 'common' storage location where required data is stored, and programs or subroutines can access it, such as the current arithmetic cell concept in ATOLL. This concept is easy to learn, would reduce the required data declarations in both subroutines and programs, and could be adapted to the needs of parameter passing for concurrently executing procedures.

The use of macros was also studied during this program conversion. Macros are a programming aid that are generally unfamiliar to a test engineer. Most current test and checkout programs are brute force procedures, that lack the required repetition of complex sequences that lend themselves to macro usage. In fact, most areas studied as candidates for macro implementation usually turned out to be more properly included as part of the executive or operating system.

5.3.1.4 <u>Table Usage</u>. Much time was spent trying to effectively use tables to perform complex tasks. These included large tables to be used similar to profile tables, varying to small tables containing only two rows. In general, table techniques developed were not altogether successful. For example, two general methods exist for determining if a switch is in auto.

```
$** TABLE DECLARE METHOD FOR SWITCH POSITION TESTING;

         DECLARE STATE TABLE (S1 HYD SYS 1 SW) WITH 2 ROWS AND 3 COLUMNS
          TITLED      (OFF)    ,   (AUTO)    ,   (ON)    WITH ENTRIES

         <LDI1572>    OFF      ,    OFF      ,    ON     ,
         <LDI1573>    ON       ,    OFF      ,    OFF    ;


STEP2110 VERIFY (S1 HYD SYS 1 SW) FUNCTIONS ARE EQUAL TO (AUTO)
         ELSE DISPLAY EXCEPTIONS
         (*S1 HYD SYS 1 SWITCH NOT IN AUTO) TO <CRT-0>;



$** COMPOUND METHOD FOR SWITCH POSITION TESTING, NO DECLARATIONS NEEDED;

STEP2110 VERIFY <LDI1572> , <LDI1573> IS OFF ELSE DISPLAY EXCEPTIONS
         (S1 HYD SYS 1 SWITCH NOT IN AUTO) TO <CRT-0>;
```

It is obvious that method 2, compounding the test parameter, is shorter to code. Likewise it is easier to debug as the sensor notation is available at the equation, and no reference to the table is required. It can be said that for most operations allowing compound operations, the referencing of variables with like attributes and states should not be accomplished using table techniques.

Table techniques are more applicable where large numbers of variables are involved and the required state may vary during the program. IAED was typical of a program where the desired state of discretes in a table was modified continually throughout the program, while in IATS the required state was established only once in the program. Study of the listings will show the techniques used for each. Two points did come to light during the study of table techniques. First, the applicability of system output messages to table compares would greatly reduce the workload of producing the required messages. Secondly, the inclusion of a "don't care" initial declaration for a particular row and column could reduce the requirements for row inhibit and enable, which can possibly lead to errors if extensive program branching is undertaken. An example of this is shown at the end of the IATS conversion.

5.3.1.5 Data Banks. The general concept of the data bank tends to reduce the GOAL procedure writing workload. It became obvious during the program that further data bank refinement is desirable. In many cases it should be possible to status a function designator defined as a load, such as to check the set point of a thermostat or the level for a tank fill. Symbolic names might also be taken from the data bank at compile time, to be included in system messages outputted as the result of table compares. The data bank also might include definition of system variables to be used for the passing of analog information from program to subroutine. Finally, a hierarchy of data banks should be studied, as the usage of data bank structures is not an exact replacement for the DISA operator. Along with this, any function designator classed as a sensor should be available to any GOAL program.

5.3.1.6 System Interface. One area where definitions are necessarily unclear is the interface with the test system. Assumptions were made to expedite program conversion areas where these assumptions were made included interface with the count clock and GMT, the availability of millisecond timers, the availability of system routines for hardware devices, (i.e., switch selector), and the availability of analog communication cells. Other ATOLL operators invoked special routines that were assumed to be available in the executive system, and no special notation has been made of these.

5.3.1.7 Miscellaneous Comments. Many other minor comments concerning the GOAL language have been listed in tabular form on the GOAL Syntax Summary. These are grouped according to syntax number, and in general do not constitute significant changes to GOAL. However, many, such as the inclusion of a NO-op statement (syntax 73) and the inclusion of basic math in the relational formula (syntax 62) represent minor modifications that would simplify procedural coding. Others, such as the comments under syntax diagrams 39 and 42 are included only for completeness.

5.3.2 Overall Summary

In general, this program conversion effort established the effectiveness of GOAL in meeting program objectives. No serious flaws or concepts were uncovered during the study. The language is impressive for its self-documentation and the overall lack of ambiguity of its various operators. However, programs tend to become lengthy when written in GOAL, and considerable keypunch effort must be expended. Minor improvements, implementation of short form coding, and effective interface with the test system should establish GOAL as a language well suited to launch vehicle checkout.

GOAL SYNTAX SUMMARY

SYNTAX NO.	DIAGRAM	SUMMARY
1	ACTIVATE TABLE	Enables all or part of a table previously inhibited, allowing table to be modified somewhat similar to a profile table. In practice exact table configuration may be unclear.
2	APPLY/ANALOG	This operation is hardware and system dependent, with no counterpart in current techniques.
3	ASSIGN	Useful only for internal names, counterpart to analog LET and external reference SET and ISSUE. Perhaps combining these to one operator desirable.
4	AVERAGE	N/C
5 - 8	BEGIN DATA BANK MACRO PROGRAM SUBROUTINE	Comments in text body for macros and subroutines.
9	BINARY NUMBER	Numeric formula does not allow use of binary and other number patterns unless declared and initialized.
10	CHARACTER	ASCII
11	CHARACTER STRING	Syntax diagrams, such as this, expand size of manual & number of diagrams for user. Some reduction is desirable. Use of 'rules' could help.
12	COLUMN NAME	N/C
13	COMMENT	N/C
14	COMPARISON TEST	Comments under 44 and 62.
15	CONCURRENT	Limited operations allowed by concurrent prefix require 'tricks' to do many required tasks. Perhaps concurrent should be an allowable prefix. For example, why allow 'and' part of 'VERIFY' syntax and not 'ELSE' option.

GOAL SYNTAX SUMMARY
(Continued)

SYNTAX NO.	DIAGRAM NAME	SUMMARY
17-25	DECLARE DATA NUMERIC LIST QUANTITY LIST QUANTITY TABLE STATE LIST STATE TABLE TEXT LIST TEXT TABLE	Stating number of rows is redundant if table or list filled.
26	DELAY	OR UNTIL option rarely used, as somewhat redundant with VERIFY.
27	DIMENSION	Should be part of QUANTITY.
28	DISABLE INTERRUPT	Unsure how interrupt is defined in data bank - possibility program could establish interrupt.
29	END	Allow program/subroutine name on this card.
30	EXPAND MACRO	Macro statements programmer oriented, not natural to engineer. Few tasks so repetitive to require use. Execute option reduces self documenting feature of GOAL. Parenthesis required on macro name when defined, but not when used.
31	EXTERNAL DESIGNATOR	Parenthesis on table name of functions limits quick visibility of external items.
32	FREE DATA BANK	N/C
33	FUNCTION DESIGNATOR	Special characters, other than '-' are unusual in names of this sort.
34	GO TO	NO-OP statement, Desirable for target.
35	HEXADECIMAL NUMBER	N/C
36	INDEX NAME	N/C
37	INHIBIT TABLE	Needed due to type of table operation, comments under 1 and in text.

GOAL SYNTAX SUMMARY
(Continued)

SYNTAX NO.	DIAGRAM NAME	SUMMARY
38	INTEGER NUMBER	N/C
39	INTERNAL NAME	Not sure of grouping for table.
40	ISSUE DIGITAL PATTERN	Confusing operator, external designator may be a state, etc. Feel issue/set/apply/let equal should be reevaluated.
41	LEAVE	Redundant to perform subroutine/program?
42	LET EQUAL	EQUAL TO option reads LET (ABC) EQUAL TO 5. Reevaluate multiplicity of 'SET' operators.
43	LETTER	N/C
44	LIMIT FORMULA	Internal name may be a table. Binary, hex, etc., not permitted unless declared as an internal name.
45	LIST NAME	N/C
46	MACRO LABEL	N/C
47	NAME	'-' should be only symbol allowed (same as for FUNCT DESIG). 16 character names should be sufficient.
48	NUMBER	N/C
49	NUMBER PATTERN	N/C
50	NUMERAL	N/C
51	NUMERIC FORMULA	Math for number patterns not allowed. Quantity must be in parenthesis here, which is not consistent. Parenthesis around imbedded numeric formula unclear.
52	OCTAL NUMBER	N/C
53	OUTPUT EXCEPTION	Very useful instruction.

GOAL SYNTAX SUMMARY
(Continued)

SYNTAX NO.	DIAGRAM NAME	SUMMARY
54	PARAMETER	N/C
55	PERFORM PROGRAM	N/C
56	PERFORM SUBROUTINE	Parameter passing (positional data) is not natural to engineers, perhaps use of arithmetic cell technique applicable.
57	PROCEDURAL STATEMENT PREFIX	Any executable statement, including 'END' should allow statement number.
58	PROGRAM NAME	N/C
59	QUANTITY	Parenthesis required in 51 inconsistent with usage in other diagrams.
60	READ	N/C
61	RECORD	External devices (CRTs, etc.) locally declared. Display package (clear, line number, etc.) may require more sophistication. Method of providing continuous monitor of parameters (other than concurrently) may be desirable.
62	RELATIONAL FORMULA	Math may be desirable such as (A) is = (B) + 5, as when checking pressure changes or current changes during application of loads.
63	RELEASE CONCURRENT	N/C
64	REPEAT	N/C
65	REPLACE	N/C
66	REQUEST KEYBOARD	N/C
67	RESUME	Same comments as 41.
68	REVISION LABEL	N/C

GOAL SYNTAX SUMMARY
(Continued)

SYNTAX NO.	DIAGRAM NAME	SUMMARY
69	ROW DESIGNATOR	N/C
70	SET DISCRETE	See comments syntax diagram 3.
71	SPECIFY	N/C
72	STATE	N/C
73	STEP NUMBER	6 digits desirable, 4 step, 2 substep, as step numbers useless for documentation unless sequential. NO-OP operator desirable as target statement.
74	STOP	Restart labels and description will be formatted on CRT, as to what each restart label accomplishes is meaningless to an operator without this description. Thus display portion redundant, and only way to delete this output is to use unrestricted restart.
75	SUBROUTINE NAME	Revision label option.
76	SYMBOL	See Syntax 10.
77	TABLE NAME	Can be external designator.
78	TERMINATE	N/C
79	TEXT CONSTANT	Use of parenthesis confusing.
80	TIME PREFIX	Implementation dependent, see text.
81	TIME VALUE	N/C
82	USE DATA BANK	Data Bank not total replacement for DISA Operator. Hierarchy of Data Banks should be studied. Sensor Data Bank should be universal. Should be able to read value of a 'load' if in table form in computer. Need analog communication cells between programs.

SECTION 6.0

SYSTEM MACROS

I

6.1 INTRODUCTION

Several 'system' type macros were written during the course of this study for the purpose of establishing macro candidate types for inclusion in the system data bank. This investigation was centered around two classes of Macros. The first type of macro was used to provide a simple interface between procedural requirements and the operational characteristics of Test hardware. The second type was used to ease the coding burden for long and/or repetitious tasks. The macros written during this investigation are included with this report.

6.2 OBJECTIVES

The objective of this investigation was to provide an overview of the requirements for macro routines to be included in the GOAL system data bank. It was considered desirable to establish those areas where macros would have the greatest applicability, and to investigate the interface between these macro types and the test engineer writing the procedure. The analysis was performed by outlining and coding into macros certain tasks performed during Apollo/Saturn checkout. These tasks included repetitious steps performed by automated procedures during vehicle checkout, and specific system responses to Atoll operators that might not be apparent to the test engineer. The application of specific system macros is dependent upon the final definition of the test system, so generation of final or 'operational' macros was not attempted.

6.3 APPROACH

The basic objective in creating the GOAL language was to define a flexible test engineer oriented language to provide ground checkout of space vehicles. The wide range of attributes and requirements of this language have been discussed elsewhere. However, one area that can lead to efficient programming in the GOAL language and reduce the coding effort is the judicious application of programming aids, such as macros. Macro techniques are not unique to GOAL, but are included in most programmer oriented languages. The applicability of macros to the engineering oriented language, GOAL, was studied from several aspects.

First, macros by themselves do not increase the capabilities of the language. What they do accomplish is to ease the burden on the test writer by reducing coding requirements. It is also natural that with this ease on coding burden the test engineer will be better able to produce a better test program.

Second, other programming aids exist to ease this coding workload which must be considered. The REPEAT statement in GOAL is an example of this. REPEAT allows segments of a GOAL program to be executed over and over again, regardless of their location in the overall program. This is quite valuable, as repetitive portions of the program need not be recopied, reducing both programmer workload and program object code size. REPEAT, when used with index variables, even allows calculation of different parameters on each pass, as may be required for testing of five separate engines. Macros were thus studied as to their relative applicability when compared to other techniques.

Thirdly, many tasks that can be handled by macros can also be handled by subroutines or subprograms. This subject is one that is discussed in many programming texts, and in GOAL the tradeoff between each method does not lend itself to a simple answer.

Finally, several tasks considered as candidates for macro subroutines must also be considered as to their inclusion in the system executive program package.

6.4 SUMMARY

Two distinct areas for potential system macro application were denoted during this investigation. First, a class of macros was generated to assist with the interface of specialized test hardware requirements with the overall test system. Secondly, a class of macros was generated that can be defined solely as programming aids.

6.4.1 Hardware Interface Macros

During vehicle checkout, the test system must interface with many types of specialized hardware items. One example of this is the switch selector. Currently ATOLL programs use a single operator to issue a switch selector channel command. The operating system then invokes a special procedure to carry out the required command and associated system validation. This procedure was coded as a macro, and is included as example 6-1.

Another area of hardware interface that is applicable to macro programming is the repetitive powering on and off of similar systems. The various auxiliary and stage power supplies are typical candidates for this application. The power supplies included have different voltage and load current capacities, but common macros can be generated. Macros covering power on, power off, and the backup battery switchover test are included as examples 6-2, 6-3, and 6-4.

6.4.2 Programming Aid Macros

A second class of macros investigated included those that were specifically written to reduce programming workload. These macros were aimed directly at the coding tasks that were so repetitive in nature that relief was suggested.

The current LCC firing room configuration requires that most switches communicating with the RCA-110A be three way switches. This allows for manual as well as automated control of the test item. However, the position of the switch can no longer be determined by a simple on-off test, and a table declaration is suggested. A simple macro was generated to perform this declaration task, and is included as example 6-5.

Another programming aid macro was generated to assist with bookkeeping. Many times in testing it is desirable to specifically note exception items in a table and display the results on request. The procedure to accomplish this is not difficult, but a system type macro was written to allow easy insertion of this technique in the program as example 6-6.

6.5 CONCLUSIONS

System macros were generated for several programming functions. There is no question that the macros generated do ease the test engineer's burden. These macros also establish that, once the engineer becomes familiar with the parameter passing, macros should become a useful tool.

Macros were especially applicable when used for the task of hardware interface. They should be applicable to the task of power application and removal of many types of hardware systems. Macros should also be useful in providing the interface between the test program and specialized hardware. They are especially advantageous for this application as they are not included as part of the system executive, and thus lend themselves to testing at other locations, as well as adapting to local configuration control.

Macros intended primarily as programming aids can be just as useful. One task already identified is the reduction in coding requirements for the test engineer, and macros can play a significant role in that reduction.

In summary, the proper use of macros should lead to more efficient and precise checkout procedures. Macros should also lead to more program standardization, as tasks will be performed in an identical manner regardless of the individual test engineer.

One test that cannot be completed at this time is a tradeoff between various programming techniques. Macros, subroutines, the REPEAT statement, and specialized system executive programs are all logical candidates when considering techniques for more effective and efficient GOAL programming. The studies required to complete this task must await further definition.

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;

$ THE FOLLOWING ARE SELECTED APPLICATIONS OF CANDIDATE TYPE SYSTEM      ;
$ MACROS.  THESE DO NOT CONSTITUE RECOMMENDED APPLICATIONS, BUT         ;
$ SERVE TO ILLUSTRATE THOSE AREAS WHERE MACRO TECHNIQUES MAY BE         ;
$ USEFUL.  THESE MACRO APPLICATIONS ARE REPRESENTATIVE OF CURRENT       ;
$ CHECKOUT PROCEDURES AND DO NOT REPRESENT EXTRAPOLATIONS AS TO         ;
$ FUTURE CHECKOUT TECHNIQUES;

$ IN GENERATING THESE MACROS BRANCHING WAS KEPT TO AN ABSOLUTE          ;
$ MINIMUM, TO REDUCE THE NEED OF PASSING UNIQUE STEP NUMBERS AS         ;
$ PARAMETERS.;




$ EXAMPLE 6 - 1   SWITCH SELECTOR CHANNEL ISSUE;

$ THE FOLLOWING IS AN EXAMPLE OF A HARDWARE ORIENTED MACRO APPLICATION.;
$ THE FUNCTIONS PERFORMED IN THIS MACRO ARE NOW PERFORMED AS PART OF   ;
$ THE SYSTEM EXECUTIVE.  HARDWARE ORIENTED MACROS SUCH AS THIS BECOME ;
$ ATTRACTIVE WHEN INFREQUENTLY USED HARDWARE FUNCTIONS ARE ENCOUNTERED,;
$ NOT JUSTIFYING INCLUSION IN THE SYSTEM EXECUTIVE, OR IF SPECIALIZED  ;
$ HARDWARE TEST REQUIREMENTS MUST BE MET AT VARIOUS LOCATIONS OR UNDER ;
$ DIFFERENT TEST SYSTEM EXECUTIVES;


 BEGIN MACRO SWITCH SELECTOR ISSUE , STAGE , CHANNEL , RECYCLE , ERROR;

$** THIS MACRO REQUIRES FOUR  ENTRIES:;

$        1. STAGE SELECTED, SIB, SIVB, SIU;
$        2. CHANNEL TO BE ISSUED IE CHAN38;
$        3. UNIQUE STEP NUMBER FOR ERROR ROUTINE (ERROR);
$        4. UNIQUE STEP NUMBER FOR RECYCLE OPTION (RECYCLE);

$** NOTE -  UTILITY FLAG <FLG0> IS USED IN MACRO TO DETECT ERRORS;
$           UTILITY FLAG <FLG1> IS USED TO INDICATE RECYCLE;

(RECYCLE) SET <FLG1> TO ON;                      $SET RECYCLE FLAG;
          SET <FLG0> TO OFF;                     $RESET ERROR FLAG;

          VERIFY <IU SW SEL PWR> IS ON ELSE DISPLAY EXCEPTIONS
          (IU SWITCH SELECTOR POWER IS OFF) TO <CRT-0>
          AND SET <FLG0> TO ON;

          VERIFY <LVDA-ESE SW> IS OFF ELSE DISPLAY EXCEPTIONS
          (LVDA-ESE SWITCH IS IN LVDA POSITION) TO <CRT-0>
          AND SET <FLG0> TO ON;
```

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;
$ EXAMPLE 6 - 1   SWITCH SELECTOR CHANNEL ISSUE (CONT) ;


         VERIFY  <SW SEL MANUAL SELECT> IS OFF ELSE DISPLAY EXCEPTIONS
         (SWITCH SELECTOR IS IN MANUAL MODE) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <(STAGE) SW SEL INHIBIT> IS OFF ELSE DISPLAY EXCEPTIONS
         ((STAGE) SWITCH SELECTOR INHIBIT IS ON) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <(STAGE) SW SEL PWR> IS ON ELSE DISPLAY EXCEPTIONS
         ((STAGE) SWITCH SELECTOR POWER IS NOT ON) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <(STAGE) SW SEL ALL ZEROS> IS ON ELSE DISPLAY EXCEPTIONS
         ((STAGE) SWITCH SELECTOR ALL ZEROS NOT ON) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <FLGO> IS OFF ELSE GO TO (ERROR);     $ERROR-NO ISSUE  ;

$** SWITCH SELECTOR CHANNEL ISSUE ROUTINE;

         SET <(STAGE) STG SELECT> TO ON FOR 100 MSEC;
         WAIT 40 MSEC;

         VERIFY <(STAGE)  STAGE SELECT> IS ON ELSE DISPLAY EXCEPTIONS
         ((STAGE) SWITCH SELECTOR STAGE SELECT FAILED) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <FLGO> IS OFF ELSE GO TO (ERROR);

         SET  <SW SEL (CHANNEL)> TO ON FOR 100 MSEC;
         WAIT 50 MSEC;

         VERIFY <SW SEL ERROR> IS OFF ELSE DISPLAY EXCEPTIONS
         (SWITCH SELECTOR ERROR DURING ISSUE OF (CHAN)) TO <CRT-0>
         AND SET <FLGO> TO ON;


$** ERROR DETECTION ROUTINE;
(ERROR)  VERIFY <FLG1> IS ON ELSE SET <FLGO> TO OFF;
         SET <FLG1> TO OFF;

         VERIFY <FLGO> IS OFF ELSE DISPLAY EXCEPTIONS
         (SWITCH SELECTOR FUNCTION NOT ISSUED FOR ABOVE REASONS)
         TO <CRT-0>;

         VERIFY <FLGO> IS OFF ELSE DISPLAY EXCEPTIONS
         (ENTER (RECYCLE) TO RETRY, (ERROR) TO CONTINUE)
         TO <CRT-0>;

         VERIFY <FLGO> IS OFF ELSE STOP AND INDICATE RESTART LABELS
         (RECYCLE), (ERROR);

END MACRO;
```

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;

$ EXAMPLE 6 - 2   AUXILLIARY AND STAGE POWER SUPPLY ON;

$ THIS MACRO IS AN EXAMPLE OF A TYPICAL POWER ON PROCEDURE THAT CAN     ;
$ BE HANDLED BY A SYSTEM MACRO.  THE ADVANTAGE TO USING A MACRO IS      ;
$ THE SPEED OF EXECUTION IF ONLY ONE OR TWO SUPPLIES ARE TURNED ON      ;
$ IN A GIVEN PROCEDURE.  IF SEVERAL SUPPLIES ARE MANIPULATED,           ;
$ SUBROUTINES MIGHT BE ADVANTAGEOUS TO REDUCE OBJECT SIZE.              ;

BEGIN MACRO  POWER ON  , (SUPPLY) , (ERROR);
$  PARAMETERS PASSED TO THIS MACRO;
$        1. SUPPLY - POWER SUPPLY DESIG IE 6D100;
$        2. ERROR - UNIQUE STEP NUMBER FOR BRANCH ON ERROR COND;

         SET <FLGO> TO ON;                             $SET ERROR FLAG;

         VERIFY <(SUPPLY) CB1> IS ON ELSE DISPLAY EXCEPTIONS
         ((SUPPLY) MAIN CIRCUIT BREAKER IS OPEN) TO <CRT-0>
         AND GO TO (ERROR);

         SET <(SUPPLY) START SW> TO ON FOR .5 SEC;

         VERIFY <(SUPPLY) SPLY ON I> IS ON WITHIN 1 SEC ELSE DISPLAY
         EXCEPTIONS ((SUPPLY) POWER SUPPLY DID NOT COME ON)
         TO <CRT-0> AND GO TO (ERROR);

$** VOLTAGE ADJUST ROUTINE, VOLTAGE JUST ABOVE SET WITH NO LOAD;

         VERIFY <(SUPPLY) VOLTS> IS LESS THAN (SET VOLTS)
         ELSE SET <(SUPPLY) VOLTS DECRS> TO ON;

         WAIT 30 SECS OR UNTIL <(SUPPLY) VOLTS> IS LESS THAN
         (SET VOLTS);

         SET <(SUPPLY) VOLTS DECRS> TO OFF;

         VERIFY <(SUPPLY) VOLTS> IS LESS THAN (SET VOLTS)
         ELSE DISPLAY EXCEPTION ((SUPPLY) VOLTAGE CANNOT BE SET)
         TO <CRT-0> AND GO TO (ERROR);

         SET <(SUPPLY) VOLTS INCRS> TO ON;

         WAIT 30 SECS OR UNTIL <(SUPPLY) VOLTS> IS GREATER
         THAN (SET VOLTS);

         SET <(SUPPLY) VOLTS INCRS> TO OFF;

VERIFY <(SUPPLY) VOLTS IS GREATER THAN (SET VOLTS) ELSE DISPLAY
         EXCEPTIONS ((SUPPLY) VOLTAGE CANNOT BE SET) TO <CRT-0>
         AND GO TO (ERROR);
```

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;
$ EXAMPLE 6 - 2   AUXILLIARY AND STAGE POWER SUPPLY ON (CONT) ;

         SET <(SUPPLY) OUTPUT PWR SW> TO ON;
         VERIFY <(SUPPLY) OUT PWR ON I> IS ON WITHIN 1 SEC ELSE
         DISPLAY EXCEPTIONS ((SUPPLY) OUTPUT POWER NOT ON)
         TO <CRT-0> AND GO TO (ERROR);

         SET <FLGO> TO OFF;            $RESET ERROR FLAG;

(ERROR)  VERIFY <FLGO> IS OFF ELSE DISPLAY EXCEPTIONS
         ((SUPPLY) POWER ON FAILED FOR ABOVE REASONS) TO <CRT-0>
         AND SET <(SUPPLY) STOP SW> TO ON FOR 1 SEC;

         VERIFY <FLGO> IS OFF THEN DISPLAY TEXT
         ((SUPPLY) POWER SUPPLY STARTED NORMALLY) ,
         (BUS VOLTAGE IS ) ((SUPPLY) BUS V) TO <CRT-0;
END MACRO;




$ EXAMPLE 6 - 3   AUXILLIARY AND STAGE POWER BATTERY TEST;


$ THIS MACRO IS USED TO TEST THE BACKUP BATTERY SWITCHOVER FOR
$ AUX AND STAGE POWER SUPPLIES;


BEGIN MACRO BATTERY TEST (SUPPLY) , (ERROR);

$   PARAMETERS PASSED TO THIS MACRO;

$        1. SUPPLY - POWER SUPPLY DESIG IE 6D100;
$        2. ERROR - UNIQUE STEP NUMBER FOR BRANCH ON ERROR;


         SET <FLGO> TO OFF;            $RESET ERROR FLAG;

         VERIFY <(SUPPLY) OUT PWR ONI> IS ON ELSE DISPLAY EXCEPTIONS
         ((SUPPLY) POWER SUPPLY NOT ON LINE) TO <CRT-0> AND
         GO TO (ERROR);
         VERIFY <(SUPPLY) BAT VOLTS> IS GREATER THAN 24 VOLTS ELSE
         DISPLAY EXCEPTIONS ((SUPPLY) BACKUP BATTERY NOT AVAILABLE)
         TO <CRT-0> AND SET <FLGO> TO ON;

         SET <(SUPPLY) BAT EN SW> TO ON;

         VERIFY <(SUPPLY) PK2> IS ON ELSE DISPLAY EXCEPTIONS
         ((SUPPLY) BATTERY ENABLE FAILED) TO <CRT-0>
         AND SET <FLGO> TO ON;

         VERIFY <FLGO> IS OFF ELSE GO TO (ERROR);

         SET <FLGO> TO ON;               $SET ERROR FLAG;

         SET <(SUPPLY) OUT PWR ON SW> TO OFF FOR 1 SEC;

         VERIFY <(SUPPLY) BATTERY ON I> IS ON ELSE DISPLAY EXCEPTIONS
         ((SUPPLY ) BATTERY SWITCHOVER FAILED) TO <CRT-0> AND
         GO TO (ERROR);

         SET <(SUPPLY) RESET SW> TO ON FOR 1 SEC;

         VERIFY <(SUPPLY) BATTERY ON I> IS OFF ELSE DISPLAY
         EXCEPTIONS (BATTERY WOULD NOT RESET) TO <CRT-0> AND
         GO TO (ERROR);

         SET <FLGO> TO OFF;              $RESET ERROR FLAG;

 (ERROR) VERIFY <FLGO> IS OFF ELSE DISPLAY EXCEPTIONS
         ((SUPPLY) BATTERY SWITCHOVER TEST FAILED FOR ABOVE REASONS)
         TO <CRT-0>;
 END MACRO;
```

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;


$ EXAMPLE 6 - 4   AUXILLIARY AND STAGE POWER SUPPLY OFF;
$ THIS MACRO TURNS OFF A TYPICAL POWER SUPPLY.  AS IN OTHER MACROS;
$ BRANCHING HAS BEEN KEPT TO A MINIMUM TO REDUCE STEP NUMBER ;
$ PARAMETER PASSING;

BEGIN MACRO POWER OFF  , (SUPPLY) , (ERROR);

$    PARAMETERS PASSED TO THIS MACRO;
$          1. SUPPLY - POWER SUPPLY TO BE TURNED OFF IE 6D100;
$          2. ERROR - UNIQUE STEP NUMBER FOR BRANCH ON ERROR COND;

           SET <FLGO> TO OFF;              $SET ERROR FLAG;

           VERIFY <(SUPPLY) AMPS> IS LESS THAN 5 AMP ELSE DISPLAY
           EXCEPTIONS (LOAD STILL APPLIED - POWER OFF NOT ATTEMPTED)
           TO <CRT-O> AND SET <FLGO> TO ON;

           VERIFY <(STAGE) LOCKUP EN> IS OFF ELSE DISPLAY EXCEPTIONS
           (LOCKUP ENABLE IS ON - POWER OFF NOT ATTEMPTED)
           TO <CRT-O> AND SET <FLGO> TO ON;

           VERIFY <(SUPPLY) BATTERY EN> IS OFF ELSE DISPLAY EXCEPTIONS
           ((SUPPLY) BATTERY ENABLE ON - POWER OFF NOT ATTEMPTED)
           TO <CRT-O> AND SET <FLGO> TO ON;

           VERIFY <FLGO> IS OFF ELSE GO TO (ERROR);

           SET <FLGO> TO ON;               $SET ERROR FLAG;

           SET <(SUPPLY) OUT PWR SW> TO OFF;

           VERIFY <(SUPPLY) OUT PWR ON I> IS OFF WITHIN 1 SEC ELSE
           DISPLAY EXCEPTIONS ((SUPPLY) OUTPUT POWER WOULD NOT TURN OFF)
           TO <CRT-O> AND GO TO (ERROR);

           SET <(SUPPLY) STOP SW> TO ON FOR .5 SEC;

           VERIFY <(SUPPLY) VOLTS > IS LESS THAN 1 VOLT WITHIN 1 SEC
           ELSE DISPLAY EXCEPTIONS ((SUPPLY) DID NOT TURN OFF)
           TO <CRT-O> AND GO TO (ERROR);

           SET <FLGO> TO OFF;

   (ERROR) VERIFY <FLGO> IS OFF ELSE DISPLAY EXCEPTIONS
           ((SUPPLY) POWER COULD NOT BE REMOVED FOR ABOVE REASONS)
           TO <CRT-O>;
  END MACRO;
```

```
$ SECTION 6 EXAMPLES - SELECTED SYSTEM MACROS;

$ EXAMPLE 6 - 5  SHORT MACRO FOR TABLE DECLARATIONS;


$ THE FOLLOWING IS AN EXAMPLE OF A VERY SHORT SYSTEM MACRO TO RELIEVE   ;
$ THE TEST WRITER FROM THE BURDEN OF CODING THE RELATIVELY LONG         ;
$ STATEMENT REQUIRED TO DECLARE THE ATTRIBUTES OF THE STANDARD THREE    ;
$ POSITION SWITCH MOST USUALLY REQUIRED FOR DISCRETE CONTROL IN THE     ;
$ PRESENT LCC FIRING ROOM CONFIGURATION.                                ;


BEGIN MACRO THREE WAY SW, (SW FUNCT NAME) , <FUNCT1> , (STATE1) ,
                                            <FUNCT2> , (STATE2);

DECLARE STATE TABLE (SW FUNCT NAME) WITH 2 ROWS AND 3 COLUMNS
          TITLED     (OFF)     , (AUTO)  ,     (ON)     WITH ENTRIES
         <FUNCT1>   (STATE1)   ,   OFF    ,   (STATE2)   ,
         <FUNCT2>   (STATE2)   ,   OFF    ,   (STATE1)   ;

END MACRO;



$ EXAMPLE 6 - 6 MACRO STORAGE TABLE;


$ EXAMPLE OF MACRO SKELETON TO RECORD TEST DATA AND INDEX AUTOMATICALLY;
$ MACRO REQUIRES DECLARE MACRO AND DISPLAY MACRO TO COMPLETE           ;
$ ALL REQUIRED TASKS. ;

BEGIN MACRO DECLARE , (NAME);
$ THIS MACRO REQUIRED TO ESTABLISH STORAGE FOR DATA;

         DECLARE QUANTITY LIST ((NAME)) WITH 100 ENTRIES;
         DECLARE QUANTITY LIST ((NAME) TIME) WITH 100 ENTRIES);
         DECLARE NUMBER ((NAME) COUNT) = 0.;
END MACRO;


BEGIN MACRO STORE , (NAME) , <FUNCT> , (FULL) ;

$ THIS MACRO REQUIRES THREE PARAMETERS;
$        1. NAME - THE NAME OF THE STORAGE TABLE;
$        2. <FUNCT> - THE FUNCT DESIG TO BE ACCESSED;
$        3. FULL - THE UNIQUE STEP NUMBER FOR BRANCH IF FULL;

         LET ((NAME) COUNT) = ((NAME) COUNT) + 1;

         IF ((NAME) COUNT) IS GREATER THAN 100 THEN GO TO (FULL);

         READ <FUNCT> AND SAVE AS ((NAME)) ((NAME) COUNT);
         READ <GMT> AND SAVE AS ((NAME) TIME) ((NAME) COUNT);

  (FULL) IF ((NAME) COUNT) IS GREATER THAN 100 THEN DISPLAY TEXT
         ((NAME) TABLE IS FULL - NO FURTHER DATA MAY BE STORED)
         TO <CRT-0>;
END MACRO;
```